



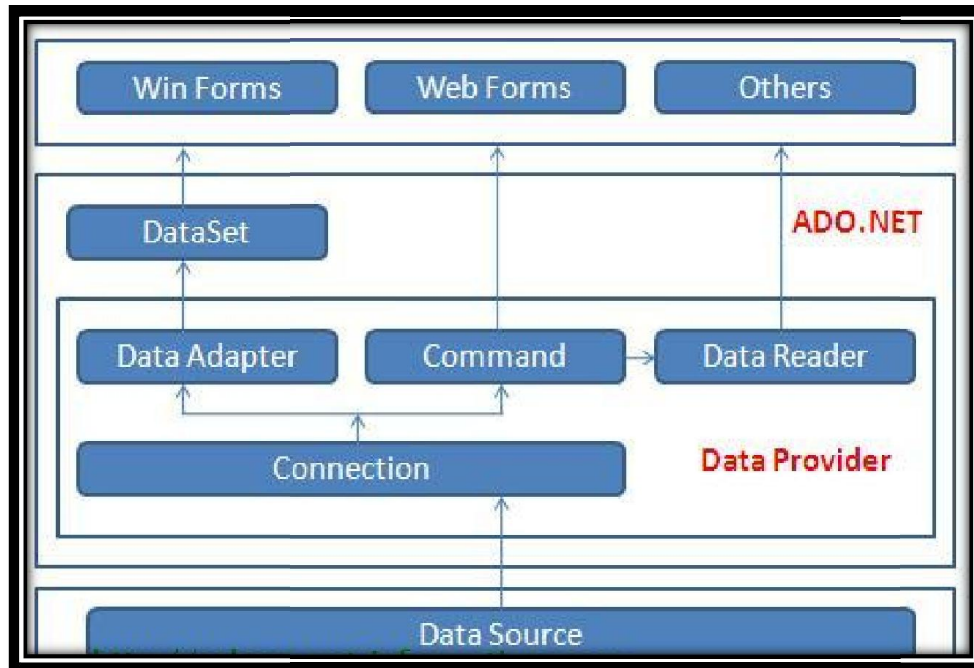
Unit : 4 Working With DataBase

Asp.net Bca Sem-5

Prepared by : Vinod.M.Makwana

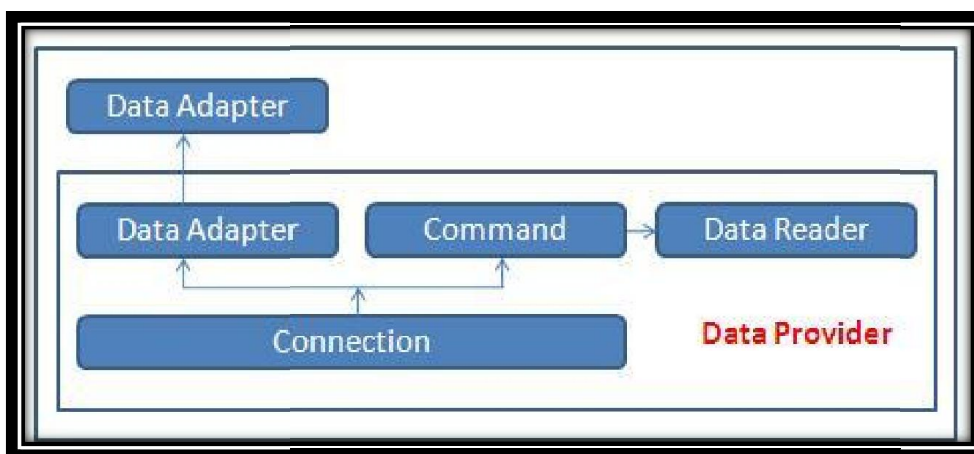
ADO.NET Architecture

ADO.NET



ADO.NET is a data access technology from Microsoft [.Net Framework](#), which provides communication between relational and non-relational systems through a common set of components. **ADO.NET** consist of a set of Objects that expose data access services to the .NET environment. ADO.NET is designed to be easy to use, and Visual Studio provides several wizards and other features that you can use to generate **ADO.NET** data access code.

Data Providers and Datasets



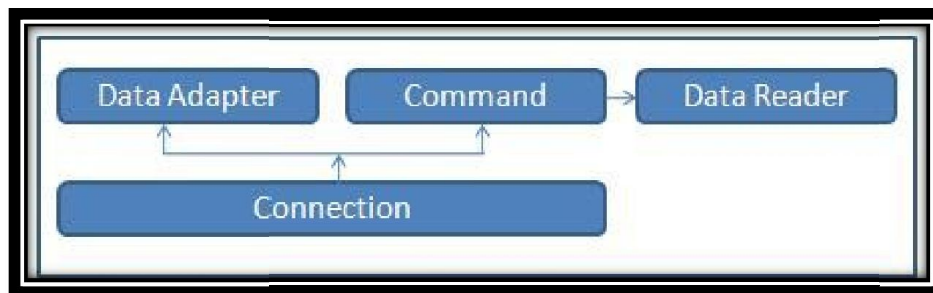
Unit : 4 Working With DataBase

The two key components of ADO.NET are **Data Providers** and **DataSet**. The .Net Framework includes mainly three Data Providers for ADO.NET. They are the Microsoft **SQL Server Data Provider**, **OleDb Data Provider** and **ODBC Data Provider**. SQL Server uses the SqlConnection object, OleDb uses the OleDbConnection Object and ODBC uses OdbcConnection Object respectively.

[C# SQL Server Connection](#)

[C# OleDb Connection](#)

[C# ODBC Connection](#)



The four Objects from the [.Net Framework](#) provides the functionality of Data Providers in the ADO.NET. They are **Connection** Object, **Command** Object, **DataReader** Object and **DataAdapter** Object. The Connection Object provides physical connection to the Data Source. The Command Object uses to perform SQL statement or stored procedure to be executed at the Data Source. The DataReader Object is a stream-based, forward-only, read-only retrieval of query results from the Data Source, which do not update the data. Finally the DataAdapter Object, which populate a DataSet Object with results from a Data Source.

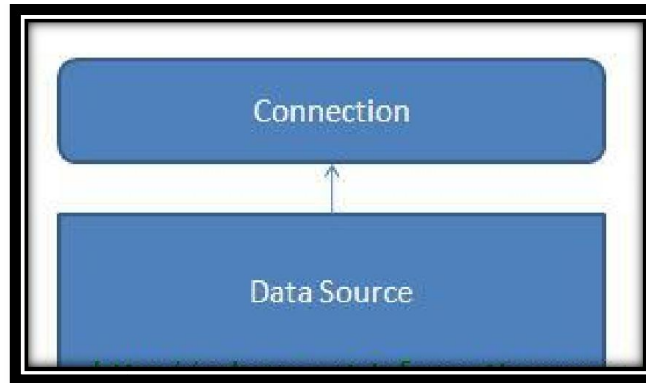
[Connection](#)

The **Connection** Object is a part of **ADO.NET** Data Provider and it is a unique session with the Data Source. The Connection Object is handling the part of physical communication between the C# application and the Data Source.

The **Connection** Object connect to the specified Data Source and open a connection between the C# application and the Data Source, depends on the parameter specified in the **Connection String**. When the connection is established, SQL Commands will execute with the help of the Connection Object and retrieve or manipulate data in the Data Source.

Once the Database activity is over, **Connection** should be closed and release the Data Source resources.

Unit : 4 Working With DataBase



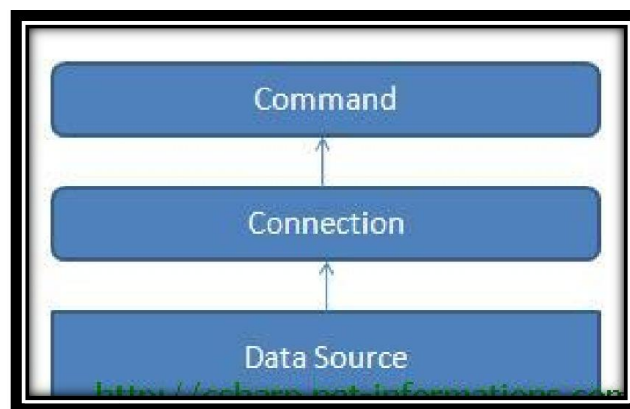
In C# the type of the Connection is depend on which Data Source system you are working with. The following are the commonly used Connections from the C# applications.

- [C# SQL Server Connection](#)
- [C# OLEDB Connection](#)
- [C# ODBC Connection](#)

C# Command

The **Command Object** in ADO.NET executes SQL statements and Stored Procedures against the data source specified in the C# Connection Object. The Command Object requires an instance of a C# **Connection Object** for executing the SQL statements .

In order to retrieve a resultset or execute an SQL statement against a Data Source , first you have to create a **Connection Object** and open a connection to the Data Source specified in the connection string. Next step is to assign the open connection to the connection property of the **Command Object** . Then the Command Object can execute the SQL statements. After the execution of the SQL statement, the Command Object will return a result set . We can retrieve the result set using a **Data Reader** .



Unit : 4 Working With DataBase

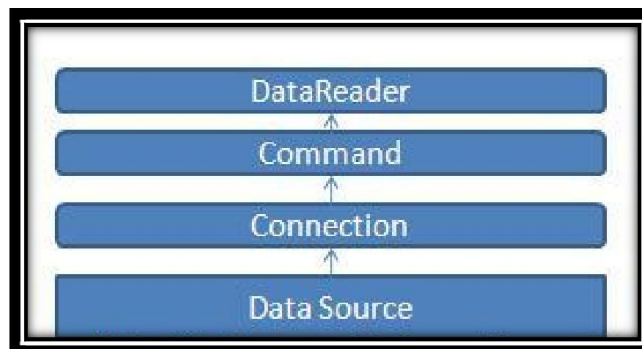
The **Command Object** has a property called **CommandText** , which contains a String value that represents the command that will be executed against the Data Source. When the **CommandType** property is set to **StoredProcedure**, the **CommandText** property should be set to the name of the stored procedure.

Click the following links to see some important built in methods uses in the Command Object to execute the SQL statements.

- [C# ExecuteNonQuery](#)
- [C# ExecuteReader](#)
- [C# ExecuteScalar](#)

C# DataReader

DataReader Object in ADO.NET is a stream-based , forward-only, read-only retrieval of query results from the Data Sources , which do not update the data. The DataReader cannot be created directly from code, they can created only by calling the **ExecuteReader** method of a Command Object.



```
SqlDataReader dr=new sqlDataReader();
```

The DataReader Object provides a connection oriented data access to the Data Sources. A **Connection Object** can contain only one DataReader at a time and the connection in the DataReader remains open, also it cannot be used for any other purpose while data is being accessed. When we started to read from a DataReader it should always be open and positioned prior to the first record. The **Read()** method in the DataReader is used to read the rows from DataReader and it always moves forward to a new valid row, if any row exist .

Unit : 4 Working With DataBase

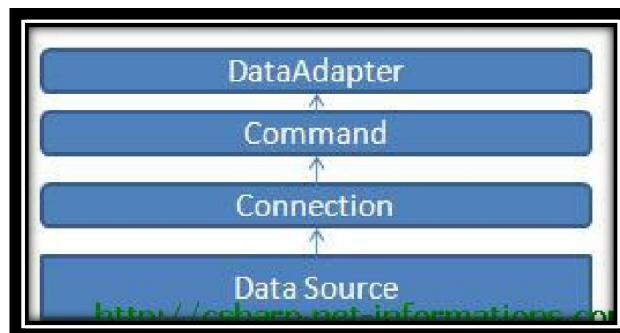
`dr.read();`

Usually we are using two types of DataReader in ADO.NET. They are **SqlDataReader** and the **OleDbDataReader**. The System.Data.SqlClient and System.Data.OleDb are containing these DataReaders respectively. From the following link you can see in details about these classes in C#.

- [C# SqlDataReader](#)
- [C# OleDbDataReader](#)

C# DataAdapter

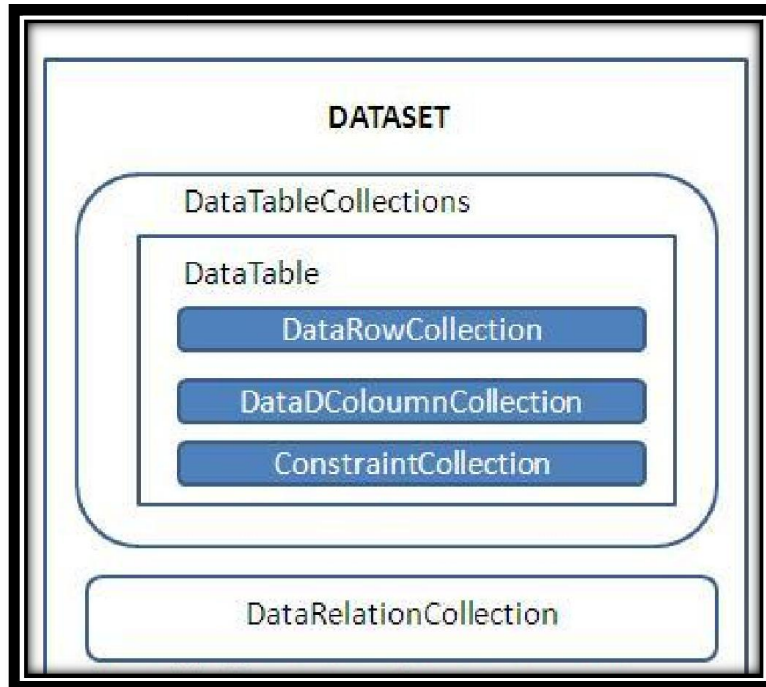
DataAdapter is a part of the ADO.NET Data Provider. DataAdapter provides the communication between the **DataSet** and the Datasource. We can use the DataAdapter in combination with the DataSet Object. DataAdapter provides this combination by mapping Fill method, which changes the data in the DataSet to match the data in the data source, and Update, which changes the data in the data source to match the data in the DataSet. That is, these two objects combine to enable both data access and data manipulation capabilities.



The **DataAdapter** can perform Select , Insert , Update and Delete SQL operations in the Data Source. The Insert , Update and Delete SQL operations , we are using the continuation of the Select command perform by 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. From the following links describes how to use SqlDataAdapter and OleDbDataAdapter in detail.

- [C# SqlDataAdapter](#)
- [C# OleDbDataAdapter](#)

DataSet



DataSet provides a disconnected representation of result sets from the Data Source, and it is completely independent from the Data Source. DataSet provides much greater flexibility when dealing with related Result Sets.

DataSet consists of a collection of **DataTable** objects that you can relate to each other with DataRelation objects. The DataTable contains a collection of **DataRow** and **DataCoulumn** Object which contains Data. The DataAdapter Object provides a bridge between the DataSet and the Data Source. From the following section you can see each of the ADO.NET components in details with **C# Source Code**.

Unit : 4 Working With DataBase

Difference between ADO and ADO.net

ADO	ADO.Net
ADO is base on COM : Component Object Modelling based.	ADO.Net is based on CLR : Common Language Runtime based.
ADO stores data in binary format.	ADO.Net stores data in XML format i.e. parsing of data.
ADO can't be integrated with XML because ADO have limited access of XML.	ADO.Net can be integrated with XML as having robust support of XML.
In ADO, data is provided by RecordSet .	In ADO.Net data is provided by DataSet or DataAdapter .
ADO is connection oriented means it requires continuous active connection.	ADO.Net is disconnected , does not need continuous connection.
ADO gives rows as single table view, it scans sequentially the rows using MoveNext method.	ADO.Net gives rows as collections so you can access any record and also can go through a table via loop.
In ADO, You can create only Client side cursor.	In ADO.Net, You can create both Client & Server side cursor.
Using a single connection instance, ADO can not handle multiple transactions.	Using a single connection instance, ADO.Net can handle multiple transactions.

Features of ADO.net

1. Bulk Copy Operation

Bulk copying of data from a data source to another data source is a new feature added to ADO.NET 2.0. Bulk copy classes provide the fastest way to transfer set of data from one source to the other. Each ADO.NET data provider provides bulk copy classes. For example, in SQL .NET data provider, the bulk copy operation is handled by SqlBulkCopy class, which can read a DataSet, DataTable, DataReader, or XML objects. Read more about Bulk Copy [here](#).

2. Batch Update

Batch update can provide a huge improvement in the performance by making just one round trip to the server for multiple batch updates, instead of several trips if the database server supports the batch update feature. The UpdateBatchSize property provides the number of rows to be updated in a batch. This value can be set up to the limit of decimal.

Unit : 4 Working With DataBase

3. Data Paging

Now command object has a new execute method called `ExecutePageReader`. This method takes three parameters - `CommandBehavior`, `startIndex`, and `pageSize`. So if you want to get rows from 101 - 200, you can simply call this method with start index as 101 and page size as 100.

4. Connection Details

Now you can get more details about a connection by setting `Connection's StatisticsEnabled` property to `True`. The `Connection` object provides two new methods - `RetrieveStatistics` and `ResetStatistics`. The `RetrieveStatistics` method returns a `HashTable` object filled with the information about the connection such as data transferred, user details, cursor details, buffer information and transactions.

5. DataSet.RemotingFormat Property

When `DataSet.RemotingFormat` is set to `binary`, the `DataSet` is serialized in binary format instead of XML tagged format, which improves the performance of serialization and deserialization operations significantly.

6. DataTable's Load and Save Methods

In previous version of ADO.NET, only `DataSet` had `Load` and `Save` methods. The `Load` method can load data from objects such as XML into a `DataSet` object and `Save` method saves the data to a persistent media. Now `DataTable` also supports these two methods.

You can also load a `DataReader` object into a `DataTable` by using the `Load` method.

7. New Data Controls

In `Toolbox`, you will see these new controls - `DataGridView`, `DataConnector`, and `DataNavigator`. See Figure 1. Now using these controls, you can provide navigation (paging) support to the data in data bound controls.

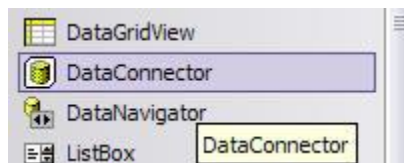


Figure 1. Data bound controls.

8. DbProvidersFactories Class

This class provides a list of available data providers on a machine. You can use this class and its members to find out the best suited data provider for your database when writing a database independent applications.

Unit : 4 Working With DataBase

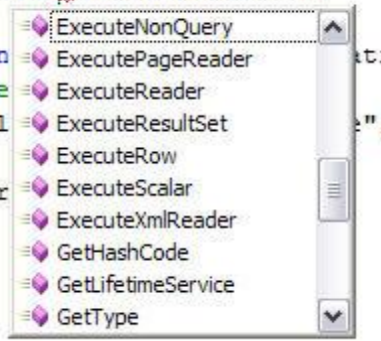
9. Customized Data Provider

By providing the factory classes now ADO.NET extends its support to custom data provider. Now you don't have to write a data provider dependent code. You use the base classes of data provider and let the connection string does the trick for you.

10. DataReader's New Execute Methods

Now command object supports more execute methods. Besides old ExecuteNonQuery, ExecuteReader, ExecuteScalar, and ExecuteXmlReader, the new execute methods are ExecutePageReader, ExecuteResultSet, and ExecuteRow. Figure 2 shows all of the execute methods supported by the command object in ADO.NET 2.0.

```
// Execute reader
SqlDataReader reader = cmd.ExecuteReader();
// Create SqlBulkCopy
SqlBulkCopy bulkData = new SqlBulkCopy(source);
// Set destination table
bulkData.DestinationTableName = "table";
// Write data
bulkData.WriteToServer(reader);
// Close objects
bulkData.Close();
destination.Close();
source.Close();
```



Features of Server Explorer

What Is the Server Explorer?

The Server Explorer is a shortcut to accessing servers, either installed on the system, or connected to the system. These servers are usually database servers such as SQL Server. By accessing the server, you access all the databases on the specific server, and then you can build the connections needed inside your program, for your program. You also can get access to SharePoint servers. I will touch on SharePoint a little bit later, but for now, I will concentrate on using the Server Explorer with an SQL Server, and then move on to the inner workings of the Server Explorer.

How Does the Server Explorer Work?

As said, the Server Explorer provides quick access to the connected database servers. With the Server Explorer, you can set up queries for use in your program. A default instance of the Server Explorer looks like the following:

Unit : 4 Working With DataBase

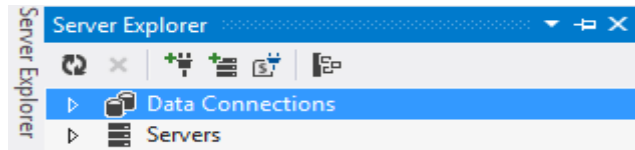


Figure 1: The Server Explorer

You can see in Figure 1 that there are three items listed:

- Data Connections
- Servers
- SharePoint Connections

Data Connections

As the name implies, the Data Connections item allows you to add, edit, and remove data connections within your program. To connect to a database on a server, follow these steps:

1. Click the Connect to Database button on the Server Explorer's toolbox.

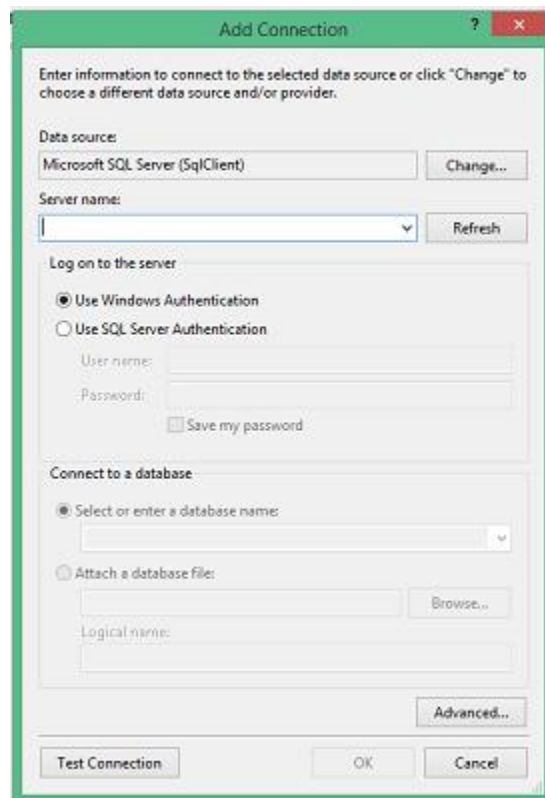


Figure 2: Add Connection

1. You simply enter (or choose) the server name, choose the database name, and click OK.
2. The connection will be entered into the Server Explorer window. It's as simple as that.

Servers

The Servers item gives you access to all the servers connected to your computer, as displayed in Figure 3:

Unit : 4 Working With DataBase

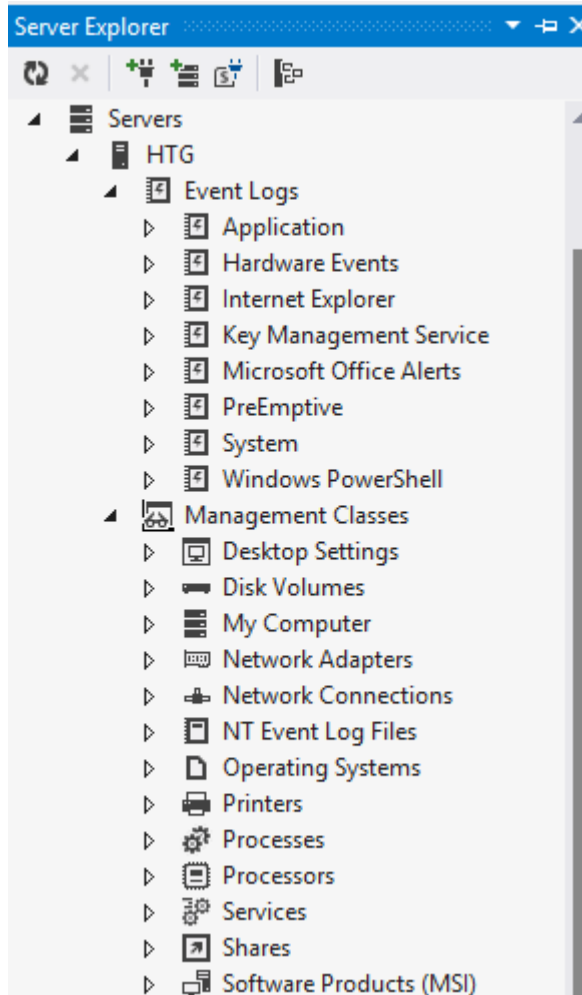


Figure 3: Servers

Apart from seeing the servers, you get access to several tools as well. These tools assist in checking the performance of your queries as well as the event logs. You can have a look here for a detailed explanation on Performance Counters.

Sql Server

Advantages of MS SQL

Installation Is Streamlined

It can be installed via a setup wizard and the prerequisite updates are detected and downloaded by the installer automatically. The complexity of installing the software is minimized significantly because of automatic installation of updates. Other components such as analytical and database services can be installed separately afterward. Automatic updation also reduces maintenance costs quite significantly.

Unit : 4 Working With DataBase

Security Features Are Better

SQL Server 2008 uses Policy-Based Management to detect security policies that are non-compliant. This feature allows only authorized personnel access to the database. Security audits and events can be written automatically to log files.

Enhanced Performance

The MS SQL server has built-in transparent data compression feature along with encryption. Users don't need to modify programs in order to encrypt the data. The MS SQL server has access control coupled with efficient permission management tools. Further, it offers an enhanced performance when it comes to data collection.

Lower Cost of Ownership

SQL server includes effective data management and data mining tools along with disk partitioning. Your server's optimum maintenance can be ensured by following effective data management practices. These practices also help you ensure the availability and recoverability of data.

Data controls

Grid View Control

ASP.NET provides a number of tools for showing tabular data in a grid, including the GridView control. It was introduced with ASP.NET 2.0. The GridView control is used to display the values of a data source in a table. Each column represents a field where each row represents a record. It can also display empty data. The GridView control provides many built-in capabilities that allow the user to sort, update, delete, select and page through items in the control. The GridView control can be bound to a data source control, in order to bind a data source control, set the DataSourceID property of the GridView control to the ID value of the data source control. It's considered a replacement for the DataGrid control from .NET 1.1. Therefore, it is also known as a super DataGrid. The GridView control offers improvements such as the ability to define multiple primary key fields, improved user interface customization using bound fields and templates and a new model for handling or canceling events. Performance is slow compared to DataGrid and ListView.

The GridView control supports the following features:

- Improved data source binding capabilities

Unit : 4 Working With DataBase

- Tabular rendering – displays data as a table
- Item as row
- Built-in sorting capability
- Built-in select, edit and delete capabilities
- Built-in paging capability
- Built-in row selection capability
- Multiple key fields
- Programmatic access to the GridView object model to dynamically set properties, handle events and so on
- Richer design-time capabilities
- Control over Alternate item, Header, Footer, Colors, font, borders, and so on.
- Slow performance as compared to Repeater and DataList control

Action	<u>Id</u>	<u>Name</u>	<u>LastName</u>	<u>Email</u>
<u>Edit</u> <u>Delete</u> <u>Select</u>	1	Mark	Arton	markarton03@outlook.com
<u>Edit</u> <u>Delete</u> <u>Select</u>	2	Tony	Jacob	tonyjacob@yahoo.com
<u>Edit</u> <u>Delete</u> <u>Select</u>	3	Robert	Brown	robertbrown@yahoo.com
<u>Edit</u> <u>Delete</u> <u>Select</u>	4	Lucas	Biglia	lucasbiglia@gmail.com

FormView control

The FormView was introduced with ASP.NET 2.0. The FormView control renders a single data item at a time from a data source, even if its data source exposes a multiple records data item from a data source. It allows for a more flexible layout when displaying a single record. The FormView control renders all fields of a single record in a single table row. In contrast, the FormView control does not specify a pre-defined layout for displaying a record. Instead, you create templates that contain controls to display individual fields from the record. The template contains the formatting, controls and binding expressions used to lay out the form. When using templates, we can place any control such as a dropdown list, checkbox and we can even place tables and rich controls like a GridView and so on. A FormView is a databound control used to insert, display, edit, update and delete data in ASP.NET that renders a single record at a time. A FormView control is similar to a DetailView in ASP.NET but the only difference is that a DetailsView has a built-in tabular rendering whereas a FormView requires a user-defined template to insert, display, edit, update and delete data.

The FormView control supports the following features:

- Template driven
- Supports column layout
- Built-in support for paging and grouping
- Built-in support for insert, edit and delete capabilities

Unit : 4 Working With DataBase

Id : 1
Name : Mark
Last Name : Arton
Email : markarton03@outlook.com
Edit New Delete
1 2 <u>3</u> 4

DetailsView control

The DetailsView control was introduced with ASP.NET 2.0. The DetailsView control uses a table-based layout where each field of the data record is displayed as a row in the control. Unlike the GridView control, the DetailsView control displays one row from a data source at a time by rendering an HTML table. The DetailsView supports both declarative and programmatic data binding. The DetailsView control is often used in master-detail scenarios where the selected record in a master control determines the record to display in the DetailsView control. It shows the details for the row in a separate space. We can customize the appearance of the DetailsView control using its style properties. Alternatively, we can also use Cascading Style Sheets (CSS) to provide styles to a DetailsView control. A DetailsView control appears as a form of recording and is provided by multiple records as well as insert, update and delete record functions.

The DetailsView control supports the following features:

- Tabular rendering
- Supports column layout, by default two columns at a time
- Optional support for paging and navigation.
- Built-in support for data grouping
- Built-in support for edit, insert and delete capabilities

Unit : 4 Working With DataBase

Id	1
First Name	Mark
Last Name	Arton
Email	markarton03@outlook.com
Delete	
New	
Edit	
1 2 3 4	

ListView Control

The ListView control was introduced with ASP.NET 3.5. The ListView control resembles the GridView control. The only difference between them is that the ListView control displays data using user-defined templates instead of row fields. Creating your own templates gives you more flexibility in controlling how the data is displayed. It enables you to bind to data items that are returned from a data source and display them. The data can be displayed in pages where you can display items individually, or you can group them. The template contains the formatting, controls and binding expressions that are used to lay out the data. The ListView control is useful for data in any repeating structure, similar to the DataList and Repeater controls. It implicitly supports the ability to edit, insert and delete operations, as well as sorting and paging functionality. You can define individual templates for each of these scenarios.

Notice that, the ListView control is the only control that is implementing the IPagedListContainer interface, so it can use a DataPager control.

The ListView control supports the following features:

- Binding to data source controls Customizable appearance through user-defined templates and styles.
- Built-in sorting and grouping capabilities
- Built-in insert, edit and delete capabilities
- Support for paging capabilities using a DataPager control.
- Built-in item selection capabilities
- Multiple key fields
- Programmatic access to the ListView object model to dynamically set properties, handle events and so on
- Fast performance as compared to GridView

Unit : 4 Working With DataBase

Id: 1 Name: Mark LastName: Arton Email: markarton03@outlook.com <input type="button" value="Delete"/> <input type="button" value="Edit"/>	Id: 2 Name: Tony LastName: Jacob Email: tonyjacob@yahoo.com <input type="button" value="Delete"/> <input type="button" value="Edit"/>	Id: 3 Name: Robert LastName: Brown Email: robertbrown@yahoo.com <input type="button" value="Delete"/> <input type="button" value="Edit"/>
Id: 4 Name: Lucas LastName: Biglia Email: lucasbiglia@gmail.com <input type="button" value="Delete"/> <input type="button" value="Edit"/>	Name: <input type="text"/> LastName: <input type="text"/> Email: <input type="text"/> <input type="button" value="Insert"/> <input type="button" value="Clear"/>	
<input type="button" value="First"/> <input type="button" value="Previous"/> <input type="button" value="Next"/> <input type="button" value="Last"/>		