DataReader

DataReader is used to read the data from database and it is a read and forward only connection oriented architecture during fetch the data from database. DataReader will fetch the data very fast when compared with dataset. Generally we will use ExecuteReader object to bind data to datareader.

To bind DataReader data to GridView we need to write the code like as shown below:

SqlConnection conn = new SqlConnection("Data Source=abc;Integrated Security=true;InitialCatalog=Test")

conn.Open();

SqlCommandcmd = new SqlCommand("Select UserName, First Name,LastName,Location FROM Users", conn);

SqlDataReadersdr = cmd.ExecuteReader();

gvUserInfo.DataSource = sdr;

conn.Close();

Holds the connection open until you are finished (don't forget to close it!).

Can typically only be iterated over once

Is not as useful for updating back to the database

DataSet

DataSet is a disconnected orient architecture that means there is no need of active connections during work with datasets and it is a collection of DataTables and relations between tables. It is used to hold multiple tables with data. You can select data form tables, create views based on table and ask child rows over relations. AlsoDataSet provides you with rich features like saving data as XML and loading XML data.

SqlConnection conn = new SqlConnection("Data Source=abc;Integrated Security=true;InitialCatalog=Test");

conn.Open();

SqlCommandcmd = new SqlCommand("Select UserName, First Name,LastName,Location FROM Users", conn);

SqlDataAdaptersda = new SqlDataAdapter(cmd);

DataSet ds = new DataSet();

da.Fill(ds);

gvUserInfo.DataSource = ds;

DataAdapter

DataAdapter will acts as a Bridge between DataSet and database. This dataadapter object is used to read the data from database and bind that data to dataset. Dataadapter is a disconnected oriented architecture. Check below sample code to see how to use DataAdapter in code:

SqlConnection con = new SqlConnection("Data Source=abc;Integrated Security=true;InitialCatalog=Test");

conn.Open();

SqlCommandcmd = new SqlCommand("Select UserName, First Name,LastName,Location FROM Users", conn);

SqlDataAdaptersda = new SqlDataAdapter(cmd);

DataSet ds = new DataSet();

da.Fill(ds);

gvUserInfo.DataSource = ds;

Lets you close the connection as soon it's done loading data, and may even close it for you automatically

All of the results are available in memory

You can iterate over it as many times as you need, or even look up a specific record by index

Has some built-in faculties for updating back to the database.

DataTable

DataTable represents a single table in the database. It has rows and columns. There is no much difference between dataset and datatable, dataset is simply the collection of datatables.

SqlConnection con = new SqlConnection("Data Source=abc;Integrated Security=true;InitialCatalog=Test");

conn.Open();

SqlCommandcmd = new SqlCommand("Select UserName, First Name,LastName,Location FROM Users", conn);

SqlDataAdaptersda = new SqlDataAdapter(cmd);

DataTable dt = new DataTable();

da.Fill(dt);

gridview1.DataSource = dt;

SQL Parameter

The SqlParameter class is found in the "System.Data.SqlClient" namespace. It is a class of a connected architecture of .NET framework. It represents parameters. To work with the SqlParameter class we should have a database.

Using parameterized queries is a three-step process:

1. Construct the SqlCommand command string with parameters.
2. Declare a SqlParameter object, assigning values as appropriate.
3. Assign the SqlParameter object to the SqlCommand object's Parameters property.

```
// 1. declare command object with parameter

    SqlCommandcmd = newSqlCommand("select * from Customers
where city = @City", conn);

// 2. define parameters used in command object

    SqlParameterparam  = new SqlParameter();

    param.ParameterName = "@City";

    param.Value         = inputCity;

// 3. add new parameter to command object

    cmd.Parameters.Add(param);
```