**In this video we will learn about**  
**1.** SqlDataAdapter  
**2.** DataSet  
  
**In the previous sessions of ADO.NET video series**, we have learnt about **SqlDataReader**. SqlDataReader is connection oriented, meaning it requires an active and open connection to the data source. SqlDataAdapter and DataSet provides us with disconnected data access model. In this part, we learn to use SqlDataAdapter and DataSet objects. In a later video session, we will discuss about, where the disconnected data access model can be used.   
  
**In Part 4, we have discussed about SqlCommand object**. When we create an instance of SqlCommand object, we pass in the following 2 parameters to the constructor of the SqlCommand class.  
**1.** The command that we want to execute   
**2.** The connection on which we want to execute the command  
  
**Along the same lines, when creating an instance of the SqlDataAdapter, we specify**  
**1.** The sql command that we want to execute   
**2.** The connection on which we want to execute the command   
  
   
  
**The example shown below**  
**1.** Creates an **instance of SqlDataAdapter**, passing in the required parameters (SqlCommandText and the Connection object)  
**2.** Creates an instance of **DataSet** object. A **DataSet** is an in-memory data store, that can store tables, just like a database.  
**3.** The **Fill()** method of the **SqlDataAdapter** class is then invoked. This method does most of the work. It opens the connection to the database, executes the sql command, fills the dataset with the data, and closes the connection. Opening and closing connections is handled for us. The connection is kept open only as long as it is needed.  
**4.** The **dataset** object, is then set as the **datasource** of the **GridView1** control  
**5.** Finally the **DataBind()** method is called, which binds the data to the control.  
string ConnectionString = ConfigurationManager.ConnectionStrings["DBConnectionString"].ConnectionString;  
using (SqlConnection connection = new SqlConnection(ConnectionString))  
{  
    // Create an instance of SqlDataAdapter. Spcify the command and the connection  
    SqlDataAdapter dataAdapter = new SqlDataAdapter("select \* from tblProductInventory", connection);  
    // Create an instance of DataSet, which is an in-memory datastore for storing tables  
    DataSet dataset = new DataSet();  
    // Call the Fill() methods, which automatically opens the connection, executes the command   
    // and fills the dataset with data, and finally closes the connection.  
    dataAdapter.Fill(dataset);  
  
    GridView1.DataSource = dataset;  
    GridView1.DataBind();  
}   
  
   
  
**Executing a stored procedure using SqlDataAdapter:**  
Create procedure spGetProductInventory   
as  
Begin  
 Select ProductId, ProductName, UnitPrice   
 from tblProductInventory  
End  
  
**If you want to execute stored procedure spGetProductInventory**, using the **SqlDataAdapter**, just specify the name of the procedure instead of the in-line sql statement.  
SqlDataAdapter dataAdapter = new SqlDataAdapter("spGetProductInventory", connection);  
dataAdapter.SelectCommand.CommandType = CommandType.StoredProcedure;  
  
**Executing a stored procedure with parameters using SqlDataAdapter:**  
Create procedure spGetProductInventoryById  
@ProductId int  
as  
Begin  
 Select ProductId, ProductName, UnitPrice   
 from tblProductInventory  
 where ProductId = @ProductId  
End  
  
**To execute stored procedure spGetProductInventoryById, we need to associate parameter**@ProductId to the SqlDataAdapeter object's SelectCommand as shown below.  
string ConnectionString = ConfigurationManager.ConnectionStrings["DBConnectionString"].ConnectionString;  
using (SqlConnection connection = new SqlConnection(ConnectionString))  
{  
    // Create an instance of SqlDataAdapter, specifying the stored procedure   
    // and the connection object to use  
    SqlDataAdapter dataAdapter = new SqlDataAdapter("spGetProductInventoryById", connection);  
    // Specify the command type is an SP  
    dataAdapter.SelectCommand.CommandType = CommandType.StoredProcedure;  
    // Associate the parameter with the stored procedure  
    dataAdapter.SelectCommand.Parameters.AddWithValue("@ProductId", 1);  
    DataSet dataset = new DataSet();  
    dataAdapter.Fill(dataset);  
  
    GridView1.DataSource = dataset;  
    GridView1.DataBind();  
}