Clases conectadas

Recuerda que tenemos dos modos de trabajo:

…..Trabajar cargando un conjunto de datos en memoria (**Clases desconectadas**)

…..Trabajar directamente con la base de datos (**Clases conectadas**).

El **objeto Command** puede trabajar con **los dos** modos de trabajo. En este apartado explicaremos cómo trabajar con los objetos Command con el **objeto DataReader** (clases conectadas).

Recordamos asimismo que el modo conectado:

... .Utiliza los objetos **Connection, Command y DataReader**.

... .Se establece una conexión permanente con el origen de datos.

....A partir de una conexión, el objeto Command generará un objeto DataReader con la información necesaria.

....Los datos del objeto DataReader son de sólo lectura.

....El objeto Command, también se encargará de realizar las operaciones de actualización con la base de datos.

En función del proveedor de datos con el que vayamos a trabajar, tenemos un determinado objeto Command:

Proveedor	**Objeto Command**
OLE DB	OleDbCommand
SQL Server	SqlCommand
ODBC	OdbcCommand
Oracle	OracleCommand

El objeto **Command** es utilizado para ejecutar sentencias SQL y Procedimientos almacenados. Los **métodos** utilizados para llevar a cabo estas tareas son:

Commando	Descripción
ExecuteReader	Ejecuta comandos que devuelven filas. Puede asignarse a un DataReader.
ExecuteNonQuery	Ejecuta comandos como instrucciones INSERT, DELETE, UPDATE y SET de Transact-SQL, que no requieren devolver filas de datos.
ExecuteScalar	Recupera un único valor de una base de datos.

Para crear un comando debemos tener una conexión creada. En los próximos ejemplos suponemos creada una conexión ***cn*** a una base de datos SQL Server, por ello el objeto command utilizado es ***SqlCommand***.

Ejemplo:

```
Dim SQLString as String = "Select * from Clientes Where Codigo > 100"

Dim cmd as New SqlCommand(SQLString, cn)
```

Usamos un SQLCommand donde especificamos una conexión cn ya existente, que será usada para seleccionar los registros de una tabla clientes cuyo código sea mayor que 100. Otra forma de obtener el mismo resultado:

```
Dim SQLString as String = "Select * from Clientes Where Codigo > 100"
Dim cmd As new SqlCommand

cmd.CommandText = SQLString
cmd.Connection = cn
```

(Ver propiedades del objeto Command en la pag. siguiente)

ExecuteReader
Ejecuta sentencias SQL que devuelven filas de datos (SELECT). Devuelve un objeto de la clase DataReader.

```
'Cadena de conexión para el proveedor SQLServer
Dim cnSQL As New SqlConnection
cnSQL.ConnectionString = "Server=LUIS\SQLEXPRESS;Database=Ejemplo;" & _
                         "Integrated Security=sspi;Persist Security Info=yes;" & _
                         "User ID=LuisR;pwd="
cnSQL.Open()
'Crear una orden de recuperación
Dim miOrden As SqlCommand = New SqlCommand("SELECT * FROM Clientes", cnSQL)
'Crear el DataReader
Dim drClientes As SqlDataReader
'Ejecutar la consulta
drClientes = miOrden.ExecuteReader
```

ExecuteScalar
El método ExecuteScalar de la clase Command devuelve un único valor, resultado de una consulta con función de agregado (SUM, AVG, COUNT, MAX E MIN).

```
'Obtener el número total de registros de la tabla dbo.Clientes
miOrden = New SqlCommand("SELECT COUNT(*) FROM Clientes", cnSQL)
Console.WriteLine("Número de filas: " & miOrden.ExecuteScalar)
```

ExecuteNonQuery

Ejecuta comandos SQL que **no** devuelven filas de datos.

```
'Modificar la Ciudad del cliente con código 10101
miOrden = New SqlCommand("UPDATE Clientes SET Ciudad='Altea' WHERE IdCliente=10101", cnSQL)
miOrden.ExecuteNonQuery()
'Añadir el cliente con IdCliente 10104
miOrden = New SqlCommand("INSERT INTO Clientes " & _
                         "(IdCliente,Apellido,Nombre,Ciudad,Provincia)" & _
                         "VALUES (10104,'Ordóñez','Manuel','Tarrasa','Barcelona')", cnSQL)
miOrden.ExecuteNonQuery()
'Eliminar el cliente con código 10298
miOrden = New SqlCommand("DELETE FROM Clientes WHERE IdCliente=10298", cnSQL)
miOrden.ExecuteNonQuery()
```

Las principales **propiedades del objeto Command** son:

Propiedad	**Descripción**
Connection	Objeto Connection necesario para conectar con la fuente de datos
CommandType	Indica de qué tipo es el comando que va a ejecutar. Podemos elegir entre los siguientes tipos: **Text:** Indica que la sentencia SQL que debe ejecutar este comando se le pasará en modo texto en forma de cadena. Este tipo es con el que aparece configurado por defecto el objeto Command, por lo que no es necesario indicar esta propiedad explícitamente. **StoredProcedure:** El comando se ejecutará mediante un procedimiento almacenado en una base de datos. Más adelante veremos cómo trabajar con procedimientos almacenados. **Table:** Recibe el nombre de la tabla.
CommandText	En caso de haber seleccionado el tipo de comando "Text" (o haberlo dejado por defecto), recibe la sentencia SQL a ejecutar.
CommandTimeOut	Tiempo de espera para ejecutar un comando. Para el proveedor Oracle no está disponible.
Parameters	Los parámetros son los valores que se sustituirán en las instrucciones SQL. Más adelante veremos cómo trabajar con parámetros, ya que es el modo de trabajo más recomendable, por seguridad y por rendimiento.

Objetos DataReader

Los objetos DataReader proporcionan una de las maneras más sencillas de leer los datos retornados por un objeto Command. Permiten acceder y recorrer los registros en modo de sólo lectura y hacia delante - forward-only.

No ofrecen acceso desconectado y **no** permiten alterar o actualizar la fuente de datos original siendo usados para **obtener rápidamente datos de sólo lectura**. Ofrece pocos recursos, pero el desempeño de su función es mucho **mejor que** el ofrecido por los objetos **DataSet.**

Las propiedades y métodos más usados de los objetos DataReader son:

1. **FieldCount** - informa del número de columnas de la fila de datos actual
2. **IsClosed** - Indica si el objeto DataReader está cerrado
3. **RecordsAffected** - especifica el número de filas alteradas , excluidas o incluidas en la ejecución de una declaración SQL
4. **Item (n)** – obtiene el valor de la n-ésima columna en su formato nativo.
5. **Close** - Método que cierra el objeto
6. **GetName** - Método que retorna el nombre de la n-ésima columna.
7. **Read** - método que permite al DataReader avanzar hasta el próximo registro
8. **IsDbNull** - método que informa si la n-ésima columna posee un valor nulo.

Cómo acceder a las filas recuperadas:

- Los **métodos *Getxxx***, permiten acceder a los contenidos de las columnas. Hay un método Getxxx para cada tipo de datos (por ejemplo: **GetDateTime**, **GetDouble**, **GetInt32** o **GetString**). El parámetro del método Get es el número de la columna a la que se accede. Por ejemplo, el siguiente código lee los campos clienteID y clienteNombre desde el registro actual del DataReader utilizando el método GetString:

  Dim clienteID As String = drClientes.GetString(0)
  Dim clienteNombre As String = drClientes.GetString(1)

  Podemos también referenciar los campos de datos en el registro actual del DataReader por nombre y a continuación llamar a una función de conversión apropiada, como se muestra en el siguiente ejemplo de código:

  Dim clienteID As String = drClientes("clienteID").ToString()
  Dim clienteNombre As String = drClientes("clienteNombre").ToString()

- El **método Read** carga en el objeto DataReader la siguiente fila. Devuelve *false* si no hay más filas. Para recorrer todos los registros y mostrarlos, podemos utilizar un bucle *While*, como se muestra en el siguiente código de ejemplo:

```
Do While drClientes.Read()
        MessageBox.Show(drCientes("clienteID").ToString(), drClientes("clienteNombre").ToString())
Loop
```

- Mientras el DataReader esté en uso, la conexión asociada está ocupada sirviendo el DataReader. Por tanto, debemos llamar a *close()* para cerrar el DataReader cuando hayamos acabado de utilizarlo:

```
drClientes.close()
```

- El **método load** presente en un *dataTable* permite obtener la información directamente desde un dataReader, sin necesidad de recorrer en éste las filas una a una:

```
Dim dt as New DataTable
dt.Load(drClientes)
DataGridView1.DataSource =dt
```