

# ADO .NET en Modo Desconectado

Es poder cargar los datos y trabajar con ellos sin necesidad de tener una conexión abierta a la base de datos. La aplicación y la base de datos no están permanentemente conectados.

# ADO .NET en Modo Desconectado

ADO.Net está orientada a un modelo de trabajo desconectado del almacén de datos, al que recurrimos sólo cuando necesitamos obtener los datos para su consulta y manipulación ó cuando esos mismos datos desconectados, los hayamos modificado y tengamos que actualizarlos en la fuente de datos.

El modelo desconectado está compuesto por tres objetos:

- DataAdapter
- Dataset
- Connection

# Objeto DataSet

Es el almacén de datos por excelencia en ADO.Net, representa una copia local de la base de datos y desconectada del proveedor de datos, la cual contiene tablas y sus relaciones.

- Cada tabla contenida dentro de un objeto **DataSet**, se encuentra disponible a través de su propiedad **Tables**.
- **DataTable** representa una tabla dentro del modelo de datos. Se organiza a través de la colección Tables que depende del Dataset.
- **DataColumn** representa una columna o campo que pertenece a una tabla.
- **DataRow** representa una fila o registro, que pertenece a una tabla.

Así:

1. Nos conectamos a la base de datos.
2. Obtenemos un conjunto de datos
3. Se cierra la conexión.
3. Trabajamos siempre en memoria, agregando, modificando, eliminando datos.
4. Una vez se termina, se descartan los cambios y actualizamos con todas las modificaciones.

# Objeto DataSet

Solo nos conectamos dos veces al principio del proceso y al final para confirmar todas las operaciones. Es más rápido.

- Para crear e inicializar las tablas del DataSet es necesario usar los objetos DataAdapter.
- Al DataAdapter le pasaremos como parámetro una cadena que representa la consulta que se va a ejecutar y que rellena el Dataset.

Del dataAdapter se utiliza el método **Fill( )**, que posee dos parámetros:

- El dataset a rellenar de información y una cadena con el nombre que tendrá la tabla creada dentro del dataset, producto de la ejecución de la consulta .

**Formato:**

**DataAdapter.Fill (dataset, “nombre de la tabla”)**

Ejemplo : DataAdapter.Fill (das, “Clientes”)

# Objeto DataSet

PROPIEDAD	DESCRIPCION
Tables.Count	Cantidad de tablas dentro del Dataset
Tables.Item("tabla")	Objeto q' representa la tabla llamada "Tabla" del dataSet
Tables("tabla").Columns	Representa Las columnas de la tabla
Tables ("tabla").PrimaryKey	Contiene las columnas que rep. la llave primaria
Tables("tabla").Rows	Representan los registros de la tabla
Tables("tabla").Rows.Item(posición)	Obtiene el registro de la tabla q' está en la posición indicada.
Tables("tabla").Rows.Item(registro).item(columna)	Obtiene el valor del campo(col) del registro indicado en la tabla del dataSet

# Objeto DataSet

**Tabla : tblClientes**

<b>ID_Cliente</b>	<b>Nombre</b>	<b>Dirección</b>	<b>Teléfono</b>
<b>1</b>	<b>María Guardia</b>	<b>La Pulida</b>	<b>111 5555-5555</b>
<b>2</b>	<b>Juan P.</b>	<b>Córdoba</b>	<b>333 4444-444</b>
<b>3</b>	<b>Daniel Pérez</b>	<b>Chaco</b>	<b>222 666-666</b>

**Si la tabla estuviese cargada en un DataSet, podríamos tener :**

**Datos.Tables.Count '1**

**Datos.Tables("tblClientes").Columns.Count '4**

**Datos.Tables("tblClientes").Rows.Count '3**

**Datos.Tables("tblClientes").Rows(0).Item("Nombre")**

**María Guardia Rows.count**

# Objeto DataAdapter

- Desempeña el papel de puente entre el origen de datos y el DataSet, permitiéndonos cargar el DataSet con información de la fuente de datos y actualizar posteriormente el origen de datos con la información del DataSet.
- Esta clase tiene cuatro propiedades, que nos van a permitir asignar cada una, un objeto Command(SqlCommand u OleDbCommand) con las operaciones estándares de manipulación de datos. Estas propiedades son:
  - InsertCommand : se utiliza para insertar datos.
  - SelectCommand : ejecuta una sentencia Select de SQL.
  - UpdateCommand: realiza una modificación de los datos.
  - DeleteCommand: se utiliza para borrar filas.
    - Método Fill ( ), ejecuta la selección del SelectCommand, los datos del origen de datos se cargan en el DataSet que pasamos como parámetro.

- Ejemplo de un objeto DataSet que llenaremos con un DataAdapter:

```
Sub Main()
```

```
Dim con As New SqlConnection()
```

```
Dim ds As New DataSet()
```

```
Dim dapt As SqlDataAdapter
```

```
.....
```

```
con.Open()
```

```
cadena = "Select * from Cliente"
```

```
dapt = New SqlDataAdapter(cadena, con) 'crea el adaptador
```

```
'Utilizar el adaptador para llenar el dataset con una tabla
```

```
dapt.Fill(ds, "Cliente")
```

```
con.Close( )
```

```
'una vez desconectados, recorrer la tabla del dataset
```

```
Dim tabla As DataTable
```

```
tabla = ds.Tables("Cliente")
```

```
Dim fila As DataRow
```

```
For Each fila In tabla.Rows
```

```
'mostrar los datos mediante un objeto fila
```

```
Console.WriteLine(fila.Item("Name") & _  
" - " & fila.Item("Direccion"))
```

```
Next
```

```
End Sub
```



# Objeto DataAdapter

## Agregando Registros

### 'crea el adaptador

```
da = New SqlDataAdapter()
```

### 'crea comandos para inserción y consulta con parámetros para

### 'asignarlos al DataSet

```
Dim cmdinserta As New SqlCommand("INSERT INTO Authors" & _  
    "(Au_Id,Au_Iname) VALUES(@Au_id,@Author )", con)
```

```
da.InsertCommand = cmdinserta
```

```
da.InsertCommand.Parameters.Add(New SqlParameter("@au_id",  
                                                    SqlDbType.VarChar))
```

```
da.InsertCommand.Parameters.Add(New SqlParameter("@Author",  
                                                    SqlDbType.VarChar))
```

# Objeto DataAdapter

Dim res As Integer

**'asigna valores a los parámetros**

da.InsertCommand.Parameters("@Au\_id").Value = TxtId.Text

da.InsertCommand.Parameters("@Author").Value = TxtName.Text

**'abrir la conexion**

**con.Open()**

**'ejecutar el comando de inserción del dataadapter**

res = da.InsertCommand.ExecuteNonQuery()

**'cerrar la conexion**

**con.Close()**

Me.cargardatos()

MessageBox.Show("Registros Añadidos " & res)

# Objeto DataAdapter

```
Private Sub cargardatos()
```

```
Dim cmdconsulta As New OleDbCommand("Select * from Authors", con)
```

```
da.SelectCommand = cmdconsulta
```

```
'crear conjunto de datos
```

```
ds = New DataSet()
```

```
'Limpiar el Dataset
```

```
ds.Clear()
```

```
'Mostrar los datos en un grid
```

```
con.Open()
```

```
da.Fill(ds, "Authors")
```

```
con.Close() 'cerrar la conexion
```

```
'proceso para mostrar los datos en un DataGridView
```

# Enlace de datos a Controles. Data Binding.

**Data Binding** es el mecanismo que permite en aplicaciones con interfaz gráfica, enlazar objetos contenedores de datos con los controles de la forma, para poder realizar operaciones automáticas de navegación y edición.

## Tipos de Data Binding

- **Enlace Simple (Simple Data Binding)** : se asocia un control que puede mostrar un único dato y el objeto que actúa como contenedor de datos. Ej: textbox.
- **Enlace complejo (Complex Data Binding)**: aquí el control que actúa como interfaz, dispone de la capacidad de mostrar varios o todos los datos del objeto que contiene la información. Ej : DataGridView.

# Enlace Simple de datos a Controles

```
cadena = "Select * from Customers"
```

```
dapt = New OleDbDataAdapter(cadena, con)
```

```
con.Open()
```

**'Utilizamos el adaptador para llenar el dataset con los registros de la tabla**

```
dapt.Fill(ds, "Customers")
```

```
con.Close()
```

```
'-----
```

'una vez desconectados, se enlaza cada textbox a los atributos de la tabla y fila asociados

'aquí en particular solo muestra la primera fila

```
'-----
```

```
textbox1.Text = ds.Tables(0).Rows(0)("CustomerId").ToString()
```

```
textbox2.Text = ds.Tables(0).Rows(0)("CompanyName").ToString
```

```
textbox3.Text = ds.Tables(0).Rows(0)("ContactTitle").ToString
```

# Enlace Complejo de datos a Controles

`ds.Clear()`

`con.Open()` 'abre la conexión

**'utiliza el adaptador para llenar el ds con una tabla**

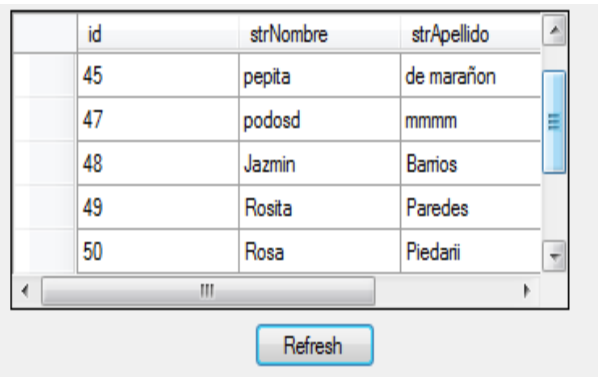
`da.Fill(ds, "Authors")`

`con.Close()` 'cerrar la conexión

**'enlazar datos con el datagrid**

`grddatos.DataSource = ds`

`grddatos.DataMember = "Authors"`



	id	strNombre	strApellido
	45	pepita	de marañon
	47	podoso	mmmm
	48	Jazmin	Banios
	49	Rosita	Paredes
	50	Rosa	Piedani

Refresh

El control que actúa como interfaz, dispone de la capacidad de mostrar varios o todos los datos del objeto que contiene la información

- Ejemplo de un objeto DataSet que llenaremos con un DataAdapter:

```
Sub Main()  
    Dim con As New SqlConnection()  
    Dim ds As New DataSet()  
    Dim dapt As New SqlDataAdapter .....  
con.Open()  
Dim dapt As New SqlDataAdapter("Select * from Cliente", con) 'crea el adaptador  
  
'Utilizar el adaptador para llenar el dataset con una tabla  
dapt.Fill(ds, "Cliente")  
con.Close()  
  
'una vez desconectados, recorrer la tabla del dataset  
Dim tabla As DataTable  
tabla = ds.Tables("Cliente")  
Dim fila As DataRow  
For Each fila In tabla.Rows  
    'mostrar los datos mediante un objeto fila  
    Console.WriteLine(fila.Item("Name") & _  
        " - " & fila.Item("Direccion"))  
Next  
End Sub
```

# Navegación y edición de registros en modo desconectado

- el objeto **DataSet**, combinado con un conjunto de objetos enfocados al mantenimiento de datos desconectados como son : el **DataAdapter**, **DataTable**, **DataRow**, nos van a permitir realizar las tareas de navegación entre los registros de una tabla del DataSet, además de la modificación de sus datos en las operaciones de inserción, modificación y limpieza de filas.



# Navegación y edición de registros en modo desconectado

## 2.1.2.3 Objeto CommandBuilder

- Su función es construir automáticamente los comandos necesarios para las operaciones de consulta, inserción, etc.
- A este objeto se le pasa como parámetro un objeto dataAdapter(contiene objetos Command para las operaciones cotidianas).

### Ejemplo:

```
Dim cmb As SqlCommandBuilder = New  
SqlCommandBuilder(da)
```

# Navegación y edición de registros en Modo Desconectado

Para recorrer la tabla es necesario obtener del DataSet, la tabla que necesitamos mediante su colección **Tables** y a la vez a la colección **Rows** de esa tabla, pasarle el número de fila/registro al que vamos a desplazarnos.

## Ejemplo :

```
Private Sub BtnProximo_Click(ByVal sender As System.Object, ByVal e As  
    System.EventArgs) Handles BtnProximo.Click  
    If (iPos = ds.Tables("Categories").Rows.Count - 1) Then  
        MessageBox.Show("Ultimo Registro")  
    Else  
        iPos += 1  
        cargardatos()  
    End If  
End Sub
```

# Operaciones de Edición

Para realizar las operaciones de edición, debemos utilizar los miembros del objeto tabla del **DataSet**. Cuando terminamos este proceso , actualizaremos el almacén de datos original con el contenido del DataSet, usando el **DataAdapter**.

## Ejemplo:

### ‘Insertar registros

```
Dim drow As DataRow
```

```
‘obtener un nuevo objeto de la fila
```

```
drow = ds.Tables("Categories").NewRow
```

```
drow("CategoryId") = Txtcodigo.Text ‘asignar valor a los campos
```

```
drow("CategoryName") = Txtnomb.Text
```

```
drow("Description") = Txtdes.Text
```

```
‘añadir la fila a la filas de la tabla del dataset
```

```
ds.Tables("Categories").Rows.Add(drow)
```

```
dapt.Update(ds, "Categories")
```

# Adición de Registros

El proceso de inserción, permite asignar los valores a las columnas del objeto DataRow.

Dim drow As DataRow

drow = **ds.Tables("Categories").NewRow**

drow("CategoryId") = 1000

drow("CategoryName") = "Tuercas"

drow("Description") = "Tres Dientes"

'añadir la fila a la filas de la tabla del dataset

**ds.Tables("Categories").Rows.Add(drow)**

# Eliminación

El proceso de eliminación es un poco diferente ya que debemos obtener la fila a eliminar mediante un objeto **DataRow** y después se borra con el **Delete**; posteriormente se actualiza la eliminación usando el método **GetChanges()** del objeto **DataTable**, obteniendo un objeto con las filas borradas, información que pasamos al **DataAdapter** para que actualice la información en la BD.

## ‘ Eliminar registros

```
Dim drow As DataRow
Dim tborra As DataTable
drow = ds.Tables("Categories").Rows(iPos)
drow.Delete() 'borrar la fila
'con el método GetChanges tabla con filas borradas
tborra = ds.Tables("categories").GetChanges(DataRowState.Deleted)
'actualizar en el almacén las filas borradas
da.Update(tborra)
ds.Tables("Categories").AcceptChanges()
```