Tema 9: ADO.NET (II)

- Modo desconectado.
- Adaptadores de datos.
- Conexiones, adaptadores y conjuntos de datos.
- La clase DataSet.
- La clase DataTable.
- La clase DataView.

Modo desconectado.

❑ En modo desconectado los adaptadores de datos se utilizan para relacionar una conexión con un conjunto de datos.

- Adapta los datos del formato nativo del gestor de bases de datos para que puedan ser utilizados en el DataSet (XML).
  - ✓ Carga las diferentes tablas en el DataSet.
  - ✓ Actualiza las modificaciones del DataSet en el origen de datos.
- Normalmente se utilizará un adaptador de datos por cada tabla que queramos recuperar del origen de datos.

❑ Crear la instancia del adaptador de datos.

- Constructor.

  Dim *nombreAdaptador* As *xxx*DataAdapter
  nombreAdaptador = New *xxx*DataAdapter(*selectSQL*, *conexión*)

  ✓ xxxDataAdapter es alguna de las clases SqlDataAdapter, OleDbDataAdapter, OdbcDataAdapter y OracleDataAdapter.

  ✓ selectSQL es una cadena con la instrucción SQL que recuperará los datos de la tabla que se añadirá al DataSet.

  ✓ conexión es un objeto Connection ya abierto.

Adaptadores de datos (II).

- Rellenar el conjunto de datos

  - El adaptador de datos permite cargar las tablas recuperadas a partir de la sentencia SQL en objetos de tipo DataTable del conjunto de datos.

  - El método Fill permite realizar la carga de datos.

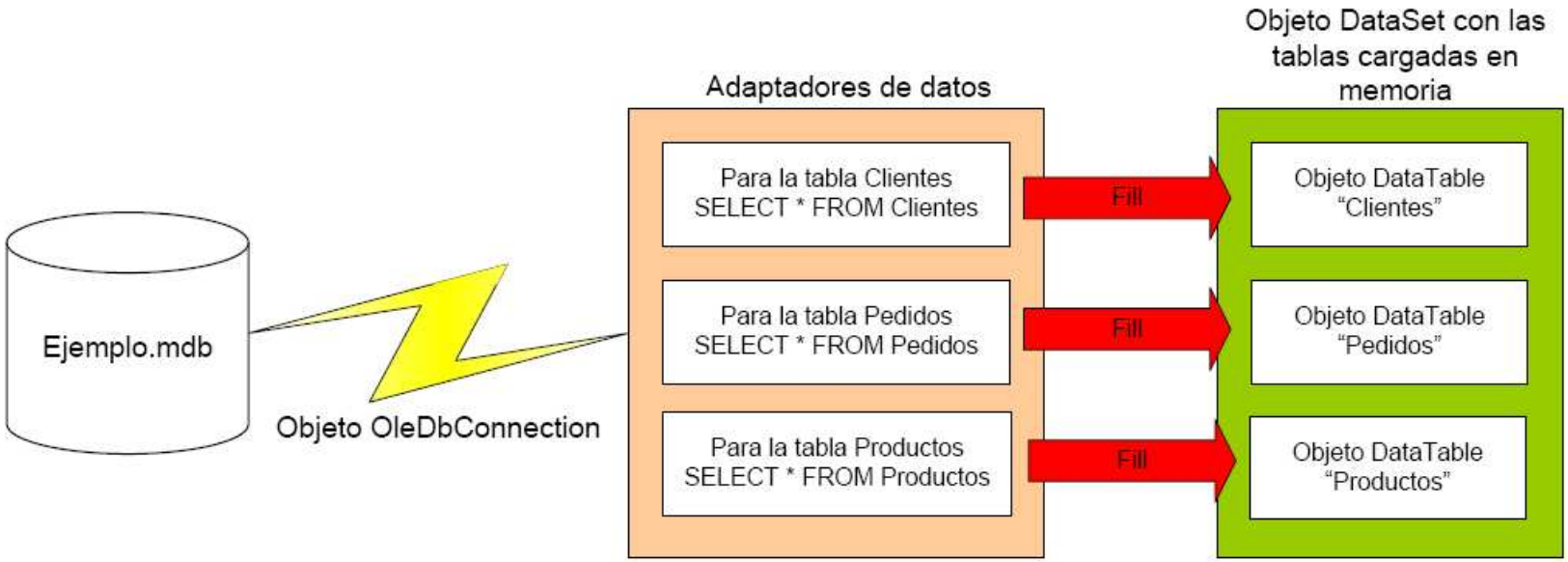
    objetoDataAdapter.Fill(*objetoDataSet*,*nombreObjetoDataTable*)

    ✓ Precisa la existencia de un objeto de la clase DataSet.

    ✓ *nombreObjetoDataTable* es una expresión de cadena que identificará la tabla dentro del conjunto de datos.

    ✓ Los datos que se cargarán en la tabla serán los que se recuperen mediante la orden SQL del constructor del adaptador de datos.

Conexiones, adaptadores y conjuntos de datos



Conexiones, adaptadores y conjuntos de datos (II)

- Se conecta a una base de datos Access y carga en el conjunto de datos las tablas Clientes y Pedidos.

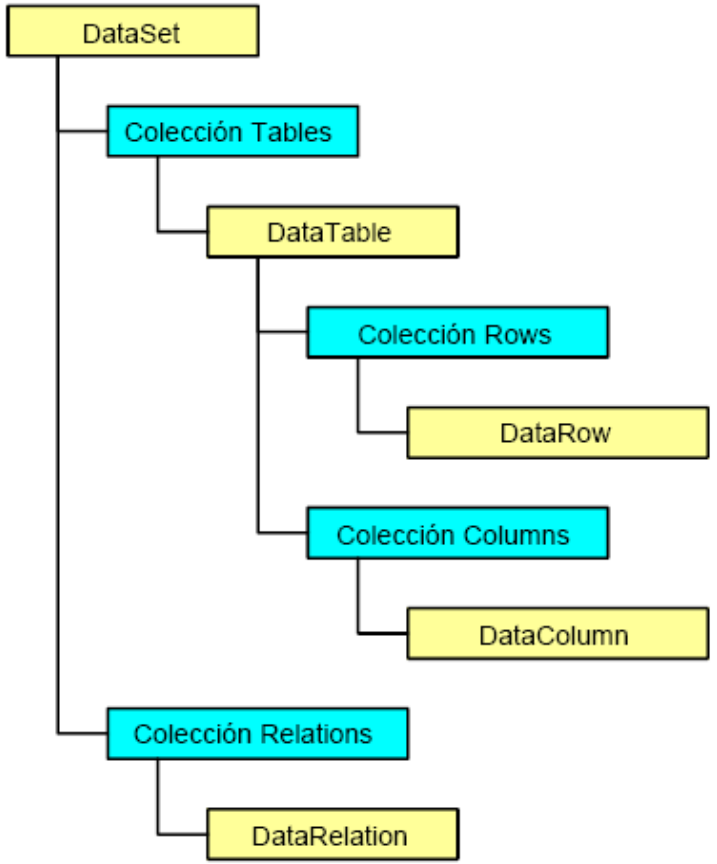
```
'Establece los espacios de nombre
Imports System.Data
Imports System.Data.OleDb
...
'Declaración de variables.
Dim cn As New OleDbConnection
Dim daClientes As OleDbDataAdapter
Dim daProductos As OleDbDataAdapter
Dim daPedidos As OleDbDataAdapter
Dim ds As New DataSet

'Cadena de conexión para el proveedor OleDB para Access
cn.ConnectionString = "PROVIDER=Microsoft.Jet.Oledb.4.0; “ & _
                      “Data Source=C:\BBDD\Ejemplo.mdb"
cn.Open()

'Crear los adaptadores de datos
daClientes = New OleDbDataAdapter("SELECT * FROM Clientes", cn)
daPedidos = New OleDbDataAdapter("SELECT * FROM Pedidos", cn)
daProductos = New OleDbDataAdapter("SELECT * FROM Productos", cn)

'Rellenar el Dataset
daClientes.Fill(ds, "Clientes")
daPedidos.Fill(ds, "Pedidos")
daProductos.Fill(ds, "Productos")

'Una vez cargado el dataset se puede cerrar la conexión
cn.Close()
```

La clase DataSet

❑ Almacena en la memoria caché del cliente los resultados de la consulta SQL establecida en un adaptador de datos.

- Los datos están disponibles en modo desconectado.

- Forman una pequeña base de datos con información necesaria para la aplicación.



La clase DataSet (II)

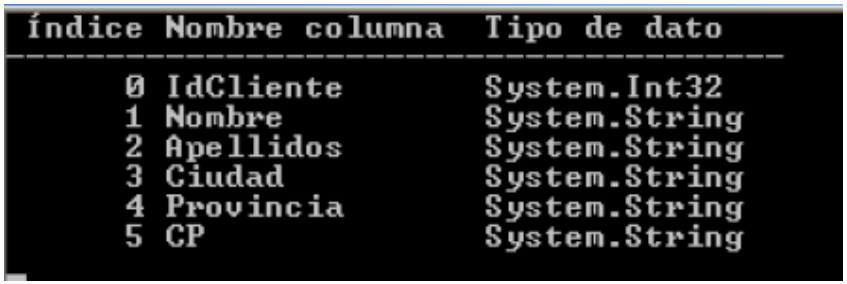
- La colección Tables.
  - Cada tabla añadida por el método Fill del adaptador de datos formará un objeto de tipo DataTable.
  - La propiedad Tables de la clase DataSet permite acceder al conjunto de objetos DataTable cargados.
  - Se puede hacer referencia a cada una de las tablas indicando el índice de la misma o mediante el nombre.
    - ✓ Para hacer referencia a la primera tabla cargada (Clientes)
      ds.Tables(0)
    - ✓ Para hacer referencia a la tabla Pedidos
      ds.Tables("Pedidos")

La clase DataSet (III)

```
'Ejemplo: obtener las tablas contenidas en el Dataset
Console.WriteLine("Tablas del DataSet")
Console.WriteLine("-------------------------")

For Each dt As DataTable In ds.Tables
    Console.WriteLine(dt.TableName)
Next
```

La clase DataTable

- Cada tabla (objeto DataTable) está formada por:
  - Una colección de columnas (colección de objetos DataColumn).
    - ✓ La propiedad Columns permite acceder a cada una de las columnas.
    - ✓ Cada columna guarda información de las características de cada uno de los campos (tipo de dato, longitud, etc.).
    - ✓ Es posible acceder a ellas mediante el índice o mediante el nombre de la columna.




```
'Obtener las columnas de la tabla Clientes
Console.WriteLine("Columnas de la tabla Clientes")
Console.WriteLine("----------------------------")
Console.WriteLine()
Console.WriteLine("{0,7} {1,-15} {2,-15}", _
"Índice", "Nombre columna", "Tipo de dato")
Console.WriteLine(New String("-", 39))
For i As Integer = 0 To ds.Tables(0).Columns.Count - 1
   Console.WriteLine("{0,7} {1,-15} {2,-15}", i, _
   ds.Tables(0).Columns(i).ColumnName, _
   ds.Tables(0).Columns(i).DataType)
Next
```

La clase DataTable (II)

❑ Una colección de filas (colección de objetos DataRow).

- La propiedad Rows permite acceder a cada una de las filas.
- Cada fila de la colección Rows está identificada por su índice (posición de la fila dentro de la colección).
  - ✓ Es posible acceder a ellas de forma similar a un array.
- El valor de cada campo está disponible mediante la colección Item del objeto DataRow.
  - ✓ Accedemos a cada campo por su índice o por el nombre del campo.

❑ Obtener la información de todas las filas de una tabla.

```
'Listado de la tabla Clientes
Console.WriteLine("Listado de la tabla Clientes")
Console.WriteLine("----------------------------")
Console.WriteLine("{0,-6} {1,-20} {2,-10} {3,-20} {4,-11} {5,-5} ", _
"Codigo", "Apellidos", "Nombre", "Ciudad", "Provincia", "CP")
Console.WriteLine(New String("-", 79))

For i As Integer = 0 To ds.Tables(0).Rows.Count – 1
  Console.WriteLine("{0,-6} {1,-20} {2,-10} {3,-20} {4,-11} {5,-5} ", _
  ds.Tables(0).Rows(i).Item("IdCliente"), _
  ds.Tables(0).Rows(i).Item("Apellidos"), _
  ds.Tables(0).Rows(i).Item("Nombre"), _
  ds.Tables(0).Rows(i).Item("Ciudad"),_
  ds.Tables(0).Rows(i).Item("Provincia"), _
  ds.Tables(0).Rows(i).Item("CP"))
Next
```

La clase DataTable (III)

```
'Listado de la tabla Productos
Console.WriteLine("Listado de la tabla Productos")
Console.WriteLine("----------------------------")
Console.WriteLine()
Console.WriteLine("{0,-11} {1,-20}", "Id.Producto", "Producto")
Console.WriteLine(New String("-", 32))

For Each dr As DataRow In ds.Tables(2).Rows
      Console.WriteLine("{0,11} {1,-20}", _
      dr.Item("IdProducto"), dr.Item("Producto"))
Next
```

Filtrar y ordenar registros en un DataTable

❑ El método Select del objeto DataTable devuelve una colección de objetos DataRow con los registros seleccionados.

- Para devolver un conjunto de filas que cumplan el filtro

  objetoDataTable.Select(*expresiónFiltro*)

- Para obtener un conjunto de filas que cumplan un filtro ordenados a partir de uno o varios campos.

  objetoDataTable.Select(*expresiónFiltro,expresiónSort)*

- *expresiónSort* es una cadena que contiene los nombres de las columnas por los que queremos ordenar separadas por comas.
  - ✓ El criterio de ordenación es ascendente. Cada nombre de columna puede ir seguido de las claves ASC o DESC para indicar el tipo de ordenación.
    - "Apellido", ordena por la columna apellidos.
    - "Apellido,Nombre", ordena por las columnas Apellido y Nombre
    - "Provincia DESC, IdCliente", ordena descendentemente por Provincia y a continuación, de forma ascendente por IdCliente.

Filtrar y ordenar registros en un DataTable (II)

❑ expresiónFiltro es una cadena con una expresión lógica que contiene el criterio de búsqueda en el siguiente formato:

nombreColumna opRelación valor

- La columna y el valor deben tener el mismo tipo de dato.
- Los operadores de relación serán =, <,>,<=,>=,<> o LIKE.
- Es posible unir varias expresiones con los operadores AND u OR.
- Para valores numéricos:

  "Precio > 100"
- Para valores de cadena, el valor se debe encerrar entre comillas simples.

  "Ciudad = '"Madrid"'
- Para valores de fecha, es necesario encerrar las fechas entre almohadillas.

  "Fecha > #23/04/2004#"
- El operador LIKE compara una columna con un patrón. En el patrón se pueden utilizar comodines como el % o el *.

  "Ciudad LIKE 'M%"'

Filtrar y ordenar registros en un DataTable (III)

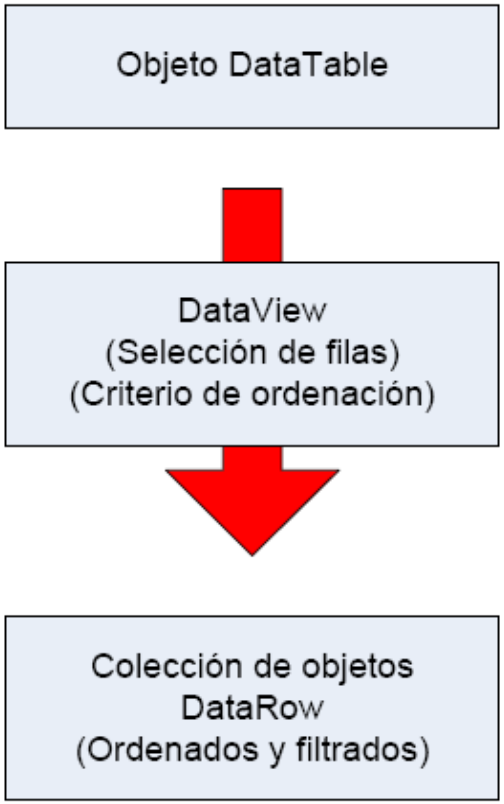
```
'Filtrar los Clientes cuya provincia empiece por M y ordenarlos por apellidos
Console.WriteLine("{0,-6} {1,-20} {2,-10} {3,-20} {4,-11} {5,-5} ", _
"Codigo", "Apellidos", "Nombre", "Ciudad", "Provincia", "CP")
Console.WriteLine(New String("-", 79))

For Each dr As DataRow In ds.Tables(0).Select("Ciudad LIKE 'M%'", "Apellidos")
   Console.WriteLine("{0,-6} {1,-20} {2,-10} {3,-20} {4,-11} {5,-5}", dr.Item("IdCliente"), dr.Item("Apellidos"), _
                     dr.Item("Nombre"), dr.Item("Ciudad"), dr.Item("Provincia"), dr.Item("CP"))
Next

'Buscar con Select un Id.Cliente introducido por teclado
Console.Write("Código cliente: ")
Dim id As String = Console.ReadLine()
Dim filas As DataRow() = ds.Tables("Clientes").Select("IdCliente=" & id)
If filas.Length = 0 Then
    Console.WriteLine("No existe")
Else
    Console.WriteLine("{0,-6} {1,-20} {2,-10} {3,-20} {4,-11} {5,-5}", filas(0).Item("IdCliente"),
                      filas(0).Item("Apellidos"), filas(0).Item("Nombre"), filas(0).Item("Ciudad"), _
                      filas(0).Item("Provincia"), filas(0).Item("CP"))
End If
```

Filtrar y ordenar registros en un DataTable (IV)

```
'Buscar con Select los clientes de una provincia introducida por teclado
Console.Write("Provincia: ")
Dim prov As String = Console.ReadLine()
For Each dr As DataRow In ds.Tables(0).Select("Provincia ='" & prov & "'")
   Console.WriteLine("{0,-6} {1,-20} {2,-10} {3,-20} {4,-11} {5,-5}", dr.Item("IdCliente"), dr.Item("Apellidos"), _
                     dr.Item("Nombre"), dr.Item("Ciudad"), dr.Item("Provincia"), dr.Item("CP"))
Next

'Buscar con Select los pedidos mayores que un precio determinado
Console.Write("Precio: ")
Dim precio As Double = Console.ReadLine()
For Each dr As DataRow In ds.Tables("Pedidos").Select("Precio >" & precio)
   Console.WriteLine("{0,-6} {1,-6} {2,-20} {3,20}", dr.Item("IdPedido"), dr.Item("IdCliente"), _
                     dr.Item("Fecha"),dr.Item("Precio"))
Next

'Buscar con Select los pedidos de una fecha determinada
Console.Write("Fecha (mm/dd/aaaa): ")
Dim fecha As String = Console.ReadLine()
For Each dr As DataRow In ds.Tables("Pedidos").Select("Fecha =#" & fecha & "#")
   Console.WriteLine("{0,-6} {1,-6} {2,-20} {3,20}",dr.Item("IdPedido"), dr.Item("IdCliente"), _
                      dr.Item("Fecha"), dr.Item("Precio"))
Next
```

La clase DataView

- Capa intermedia que se sitúa entre la tabla.
  - Proporciona una vista filtrada y ordenada de las filas de la tabla.
  - Los datos de la tabla no varía, sólo cambia el filtro y la ordenación de las filas.
  - Por su versatilidad se suele utilizar en aplicaciones que necesiten un enlace a datos.
- Permite distintas vistas de una DataTable.
  - Por ejemplo…
    - ✓ Puede servir para mostrar los datos de una tabla con distintos criterios de ordenación.
    - ✓ Pueden enlazarse a controles distintos de una aplicación para mostrar en una rejilla todas las filas y en otra las filas eliminadas.



La clase DataView (II)

- Presenta una vista dinámica de la tabla a la que está asociado.
  - Las modificaciones que se efectúen en el DataView pueden tener efecto en la tabla.
- Es similar al concepto de vista de una base de datos, pero…
  - No puede excluir ni añadir columnas de la tabla a la que está asociado
  - No puede proporcionar vistas de tablas combinadas.

La clase DataView (III)

- Construir un objeto DataView.
  - A partir de la propiedad DefaultView de la clase DataTable.

    Dim dv As DataView = ds.Tables("Clientes").DefaultView

    ✓ Proporciona una vista de todas las filas de la tabla Clientes a con su ordenación original.
  - Mediante el contructor de la clase DataView.

    DataView(*dataTable*, *expresiónFiltro*, *expresiónSort*, *dataViewRowState*)

    ✓ *expresiónFiltro* es una cadena con el criterio de selección en el mismo formato que en el método Select de la clase DataTable.

    ✓ *expresiónSort* es una cadena con la expresión de ordenación en el mismo formato que el método Select de la clase DataTable.

    ✓ *dataViewRowState* es un miembro de la enumeración DataViewRowState que indica el estado de las filas que se seleccionarán (ver diapositiva siguiente).

La clase DataView (IV)

❑ Filtrar y ordenar filas.

● La propiedad RowFilter de la clase DataView permite filtrar los registros de la vista.

✓ El valor de la propiedad sería una cadena con la expresión de filtro (igual que el método Select de la clase DataTable).

● La propiedad Sort permite ordenar las filas de la vista.

✓ El valor de la propiedad sería una cadena con la expresión de ordenación (igual que el método Select de la clase DataTable).

● La propiedad RowStateFilter es un miembro de la enumeración DataViewRowState que establece el filtro sobre el estado de la fila.

La clase DataView (V)

- Enumeración DataViewRowState

Nombre de miembro	Descripción
Added	Fila nueva.
CurrentRows	Filas actuales, incluidas las filas sin modificar, las nuevas y las modificadas.
Deleted	Fila eliminada.
ModifiedCurrent	Versión actual, que es una versión modificada de los datos originales (vea **ModifiedOriginal**).
ModifiedOriginal	Versión original (aunque se haya modificado y esté disponible como **ModifiedCurrent**).
None	Ninguno.
OriginalRows	Filas originales, incluidas las filas sin modificar y las eliminadas.
Unchanged	Fila sin modificar.

La clase DataView (VI)

- Acceso a las filas de la vista.
  - El número de filas resultante se obtiene con la propiedad Count.
  - El objeto DataView proporciona una colección objetos DataRow accesible mediante la propiedad Item.

  Podemos acceder a ellas a partir del índice de cada elemento de la propiedad Item.

  dv.Item(0).Item("IdCliente")

  Proporciona el identificador de cliente de la primera fila (fila 0) de la vista de datos.

```
'Filtrar los registros de la provincia de Madrid ordenados por Ciudad con un DataView
Dim dv As DataView = ds.Tables("Clientes").DefaultView
dv.RowFilter = "Provincia='Madrid'“
dv.Sort = "Ciudad“

For i As Integer = 0 To dv.Count - 1
   System.Console.WriteLine("{0,-6} {1,-20} {2,-10} {3,-20} {4,-11} {5,-5}", _
   dv.Item(i).Item("IdCliente"), _
   dv.Item(i).Item("Apellidos"), _
   dv.Item(i).Item("Nombre"), _
   dv.Item(i).Item("Ciudad"), _
   dv.Item(i).Item("Provincia"), dv.Item(i).Item("CP"))
Next
```