Cap. IV Acceso a Base de Datos con Código

Prof. Mitzi Murillo de Velásquez Msc.

Universidad Tecnológica de Panamá

Acceso a Base de Datos con Código

* **2.1. El Modelo ADO.Net**
* No es un simple upgrade, sino que es todo un modelo de acceso a datos cambiado. Su principal diferencia es que está diseñado para trabajar desconectado de la base de datos(sin mantener conexiones prolongadas).
* Es un intermediario entre nuestra aplicación y la base de datos.

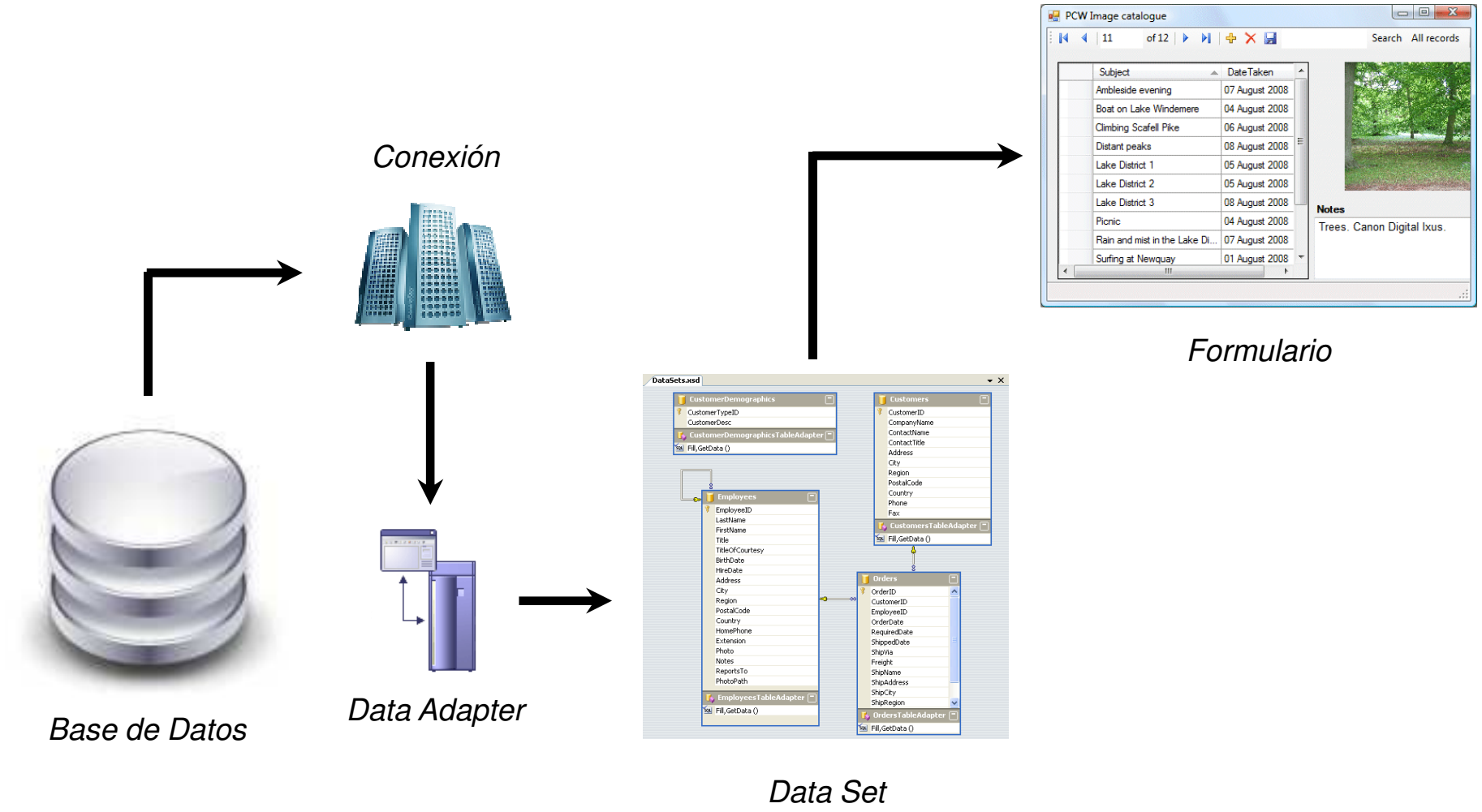
Las clases de ADO.NET

namespaces	Descripción
System.Data.	Clases comunes para trabajar con cualquier tipo de bases de datos. Ejemplo, la clase DataAdapter es heredada por las clases de cada tipo OleDbDataAdapter o SqlDataAdapter)
System.Data.OleDb	Contiene todas las clases para conectarse con bases de datos a través de OLE DB
System.Data.SqlClient	Contiene todas las clases para conectarse con bases de datos SQL Server 7.0 o posterior.

Clases de Acceso a los datos a través de un proveedor OLE DB

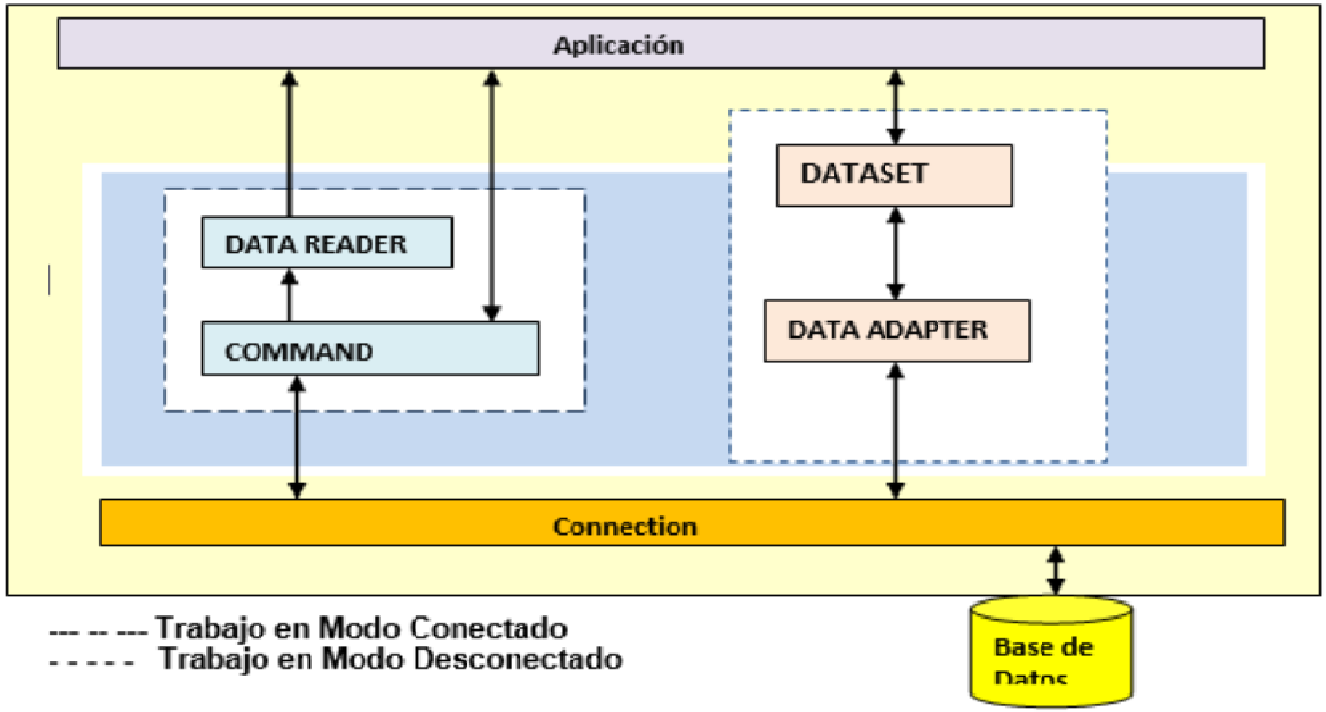
CLASE	NAMESPACE	DESCRIPCIÓN
OledbConnection	System.Data.OleDb	Establece una conexión a una fuente de datos específica.
OleDbCommand	System.Data.OleDb	Comando que se puede ejecutar sobre la base de datos (consulta ó SP)
OleDbDataReader	System.Data.OleDb	Clase que permite leer los resultados de un comando, de Forma secuencial y siempre hacia adelante.
OleDbDataAdapter	System.Data.OleDb	Agrupa una conexión y varios comandos que se ejecutarán sobre ella, sobre datos que están relacionados.
DataSet	System.Data	Contiene una representación en memoria de datos obtenidos de una base de datos. Diseñado para trabajar con independencia de la fuente de datos.

Estrategia de ADO.NET



Acceso a Base de Datos con Código

Modos de Trabajo de ADO



* Acceso a Base de Datos con Código

* **Objeto Connection**

Establece y manipula la conexión a la fuente de datos. Una conexión representa una sesión en la base de datos y a través de ella se pasan los comandos y sus resultados.

Métodos del Objeto Connection

Open: abre la conexión

Close: cierra la conexión y libera todas las fuentes asociadas.

* Acceso a Base de Datos con Código

Propiedades del objeto Connection .

ConnectionString : cadena utilizada para conectarse al origen de datos, cuando se ejecuta el método OPEN.

Provider : el nombre del proveedor de datos OLEDB.

Data Base : el nombre de la base de datos a abrir al realizar la conexión.

Server Version : Versión del servidor según la definición del proveedor de datos, (OLEDB ó SQL Server).

State: un valor de connectionState indicándole el estado actual de la conexión.

Configurando la propiedad ConnectionString:

Provider : especifica el nombre del Proveedor OLEDB para conectar los datos (SQL ó Microsoft Jet OLEDB 4.0)

Data Source : especifica la dirección de la B.D. Puede ser la ruta a una Base de datos de Access o el nombre de la máquina en la que está la B.D.

User ID/Password : nombre del usuario y la contraseña del usuario.

Initial Catalog: especifica el nombre de la BD, cuando se está conectando a una fuente de datos (SQL Server ú Oracle).

*Escoger el tipo de Conexión

Para crear la conexión, la propiedad Connectionstring usa las clases OleDbConnection ó SqlConnection en el caso de SQL Server.

Ejemplo:

```
Dim cn New As OleDb.OledbConnection
cn.ConnectionString =
"Provider=Microsoft.Jet.OLEDB.4.0;Password=;
 User ID=Admin;Data Source=C:\...Neptuno.mdb"
 cn.Open( )
  ....
  cn.Close( )
```

```
Dim SqlPubstr as String = "Data Source = (local); userID =
 sa; Initial Catalog = Pubs; Integrated Security=True"

Dim cn as New SqlConnection()
 cn.conectionstring = SqlPubstr
 cn.Open()
 'cerrar la conexión
 cn.Close()
```

*'Si es un objeto SQL Connection

```
Sub Main()
      Dim cn As New OleDb.OleDbConnection() 'objeto de tipo Conexion

     cn.ConnectionString =
      "Provider=Microsoft.Jet.OLEDB.4.0;Password=;" _
       & "User ID=Admin;Data Source = C:\Archivos de
        programa\Microsoft Visual Studio\VB98\BIBLIO.MDB  "
      Try
         cn.Open()
      Catch ex As Exception
         Console.WriteLine(ex.Message)
      End Try
      If cn.State = ConnectionState.Open Then
         MsgBox("CONEXION SATISFACTORIA")
      Else
         Console.WriteLine("NO SE PUDO ESTABLECER LA CONEXIÓN")
      End If
        Console.Read()
     cn.Close()
  End Sub
```

Objeto DataReader

- Proporciona una forma de leer una secuencia de registros de datos.
- Devuelve un flujo de datos de sólo lectura y de recorrido sólo hacia adelante.
- Para poder cargar el DataReader, aplicamos el método ExecuteReader () del objeto Command basado en una consulta SQL o procedimiento Almacenado.
- Se pueden leer tipos de DataReader:

  a) para leer conjunto de registros

  b) para leer un único registro

  c) datos en un formato XML

Objeto DataReader

Propiedades

FieldCount : devuelve el número de columnas(campos) presentes en la fila(registro)

IsClosed : devuelve los valores True o False, para indicar si el objeto DataReader está o no cerrado.

* **Item** : devuelve en forma nativa el valor de la columna cuyo nombre le indicamos como Índice en forma de cadena de texto.

Métodos

Read( ) : Desplaza el cursor actual al siguiente registro permitiendo obtener los valores del mismo a través del objeto DataReader.

Close( ). Cierra el objeto DataReader liberando los recursos.

GetXXX :permiten obtener los valores de la columnas contenidas en forma de tipo de datos GetBoolean( ), GetString( ), GetChar( ), etc.

*DataReader

Operaciones:

Para utilizarlo hace falta una conexión y un comando.

```
Dim  cn    As New SqlConnection()
Dim cmd  As  New SqlCommand()
Dim  dr   As SqlDataReader

cn.ConnectionString = “Data Source=LOCALHOST; “ & _
      “Initial Catalog=Northwind;user ID=sa”
cn .Open()
cmd = New SqlCommand( “Select * From  Publishers”, cn)
dr = cmd.ExecuteReader()
 While dr.Read
     Console.WriteLine  dr(“pub_name”) & “-“ & dr(“city”))
 End While
  dr.Close()
  cn.Close()
  Console.Read()
```

```
Call conexion()
        cadena = "Select Author,Au_ID  from Authors WHERE Au_ID = 21 "
        cmd = New OleDbCommand(cadena, cn)

        If cn.State = ConnectionState.Open Then
            MsgBox("Conexion Satisfactoria")
            Try
                dr = cmd.ExecuteReader
                dr.Read()
                TextBox1.Text = dr.Item("Au_ID")
                TextBox2.Text = dr.Item("author")
            Catch exc As Exception
                MsgBox("error  " & exc.Message)
            End Try
        Else
            MsgBox("No se pudo establecer la conexion")
        End If
        dr.Close()
        cn.Close()
    End Sub
```

*Objeto Command

Permite ejecutar sentencias Sql sobre la base de datos . Prepara las instrucciones que se van a ejecutar mas de una vez.

PROPIEDADES :

Connection : contiene la cadena con los atributos de conexión asociados al comando.

CommandType : valor que identifica el tipo de sentencia que se va a ejecutar el cual puede ser:

1. Una sentencia SQL
2. El nombre de un Procedimiento Almacenado
3. Una tabla

CommandText : establece el texto que se va a ejecutar.
Parameters : colección de parámetros asociados a este comando.

*Objeto Command

***un texto**

Comando.commandType = CommandType.Text

Comando.Commandtext = “Select * FROM tblclientes WHERE nombre LIKE ‘A%’”

***un procedimiento almacenado**

Comando.commandType = CommandType.StoredProcedure
Comando.Commandtext = “sp_TraerClientes”

***una tabla**

Comando.commandType = CommandType.tableDirect
Comando.Commandtext = “tblClientes”

Objeto Command

Métodos

* **Cancel :** cancela la ejecución del objeto Command; no se produce ningún error en el caso de que el comando no se esté ejecutando.

* **CreateParameter :** crea un objeto Parameter conectado a este comando parametrizado.

Ejemplo :

```
Imports System.Data.SqlClient
Module Module1
Sub Main()
Dim Conexion As New SqlConnection()
Dim comando As New SqlCommand()
Dim resul As SqlDataReader
    Conexion.ConnectionString = “Data Source=LOCALHOST; “ & _
    “Initial Catalog=Northwind;user ID=sa”
        Conexion.Open()
        comando.CommandText = “Select * From  Shippers”
        comando.CommandType = CommandType.Text
        comando.Connection = Conexion
        resul = comando.ExecuteReader
    While resul.Read
        Console.WriteLine(resul.Item(0).ToString & “: “ & _
         resul.Item(1).ToString)
   End While
resul.Close()
Conexion.Close()
Console.Read()
End Sub
```

Tipos de Resultados de un Comando

Los resultados se pueden obtener de 3 formas diferentes, dependiendo del método que usemos sobre el objeto comando.

ExecuteReader	ejecuta el comando y devuelve los registros en un objeto DataReader
ExecuteNonQuery	Ejecuta el comando, pero no se preocupa por devolver ningún resultado (sólo el # de registros afectados). Para SP que efectúan acción pero no devuelven resultados.
ExecuteScalar	Ejecuta un comando y devuelve el valor numérico del primer campo del registro

* Para consultas que no devuelven registros como INSERT, UPDATE o DELETE se utiliza ExecuteNonQuery

```
'comando que no devuelve registros (sólo los elimina)
 Comando.CommandText = "DELETE FROM
                       tblCLientes WHERE ID_Cliente=1"
 Comando.connection = Conexión
Comando.ExecuteNonQuery( )
```

* ‘comando que devuelve un conjunto de registros
cmd.CommandText = “Select * From Publishers”
cmd.connection = Conexión
dr = cmd.ExecuteReader()

‘comando que devuelve un solo campo, con el total de registros
Comando.CommandText = “SELECT Count (*) FROM tblCLientes ”
Comando.connection = Conexión
Comando.ExecuteScalar( )

*Objeto Command

* 3. Manejo de Parámetros

Un parámetro es una entrada de datos en una sentencia SQL, cuyo valor se proporciona a través de una variable y se inserta antes de ejecutar la sentencia.

Los objetos Parameter se pueden crear de tres maneras :

.

- **Con el constructor de Parameters,**
- **Utilizando el método CreateParameter del Command,**
- **Invocando el método AddWithValue de la colección Parameters**

1. Utilizando el constructor

```
Dim por As New OleDBParameters("PubID", OleDBType.Integer)
Por.Value =156                          'Define el  valor de parámetro
cmd.Parameters.AddWithValue(por)        'lo agrega a la colección
Dim por2 As New OleDBParameters("YearPub",OleDBType.SmallInt)
Por2.value =1993
cmd.Parameters.Add(por2)
```

2. Utilizando el método CreateParameter

```
Dim por As New OleDBParameters = cmd.CreatePameter( )
por.value=156
cmd.Parameter.Add(por)
por = cmd.CreateParameter          'reutilice la variable
por.value    = 1993
cmd.Parameters.Add(por)
```

3. Invocando el método AddWithValue

```
cmd.Parameters.AddWithValue("PubID", 156)
cmd.Parameters.AddWithValue(YearPub", 1992)
```

Todas esta formas trabajan para la siguiente consulta :

Select * From titles where PubUID = ? AND [YearPublished] = ?

OLEDBParameter usando CreateParameter

```
Dim var As Integer
        Call conexion()
        cadena = "Select Author from Authors WHERE Au_ID = ?"
        cmd = New OleDbCommand(cadena, cn)

         Dim par As OleDbParameter = cmd.CreateParameter
                var = Val(InputBox("Teclee un código"))
                par.Value = var
                cmd.Parameters.Add(par)
                dr = cmd.ExecuteReader
                dr.Read()
                TxtAutor.Text = dr.Item("author")

        dr.Close()
        cn.Close()

    End Sub
```

OLEDBDParameter con ADDWithValue

```
Dim valor As Integer
        Call conexion()
        cadena = "Select Author from Authors WHERE Au_ID = ?"
        cmd = New OleDbCommand(cadena, cn)

            valor = Val(InputBox("Teclee un código"))
              cmd.Parameters.AddWithValue("Au_Id", valor)
              dr = cmd.ExecuteReader
              dr.Read()
              TextBox1.Text = dr.Item("au_id")
              TextBox2.Text = dr.Item("author")

        dr.Close()
        cn.Close()

    End Sub
```

Con SQL Parameter

Select * FromTitles where titleID = @ titleID

con el constructor

```
Dim por as New SqlParameter (“@titleID”, SqlType.varchar)
por.Value = “BU1032”                ‘valor del parámetro
cmd.Parameters.AddWithValue(por)
```

‘Con CreateParameter

```
Dim por as SqlParameter = cmd.CreateParameter
Por.Value = “BU1032”                ‘valor del parámetro
cmd.Parameters.Add(por)‘
```

Con el mètodo ADD

```
cmd.Parameters.AddWithValue(“@TitleID”, “BU1032”)
```

SQL con Parámetros

```
Dim valor As Integer
 Call conexion()
 cadena = "Select *  from Shippers WHERE shipperid =@shipperID"
 cmd = New SqlCommand(cadena, cn)

  ‘ parametros con New
 Dim param As New SqlParameter("@shipperid", SqlDbType.Int)

 valor = Val(InputBox("Teclee un código"))
 cmd.Parameters.AddWithValue("@shipperid", valor)
 dr = cmd.ExecuteReader
 dr.Read()
TextBox1.Text = dr.Item("CompanyName")
 dr.Close()
cn.Close()
```

* Proveedores para ADO.NET

El proveedor de datos es el responsable de crear una comunicación entre la aplicación y el origen de datos, así como de suministrar todos los componentes necesarios para trabajar con los datos.

Proveedores	Descripción
.Net Framework para OleDb	Para orígenes de datos que se exponen con OLEDB. Usa el espacio de nombres System.Data.OleDB.
.Net Framework para SqlClient	Proporciona acceso a datos para Microsoft SQL Server. Usa el espacio de nombres System.Data.SqlClient.

Es un provee
para bases d
que aprovech
las caracterís
SQLServer.
La comunicac
realiza directa
intermedias (
De este mod
escalabilidad
La funcionalic
SQLServer.N

Ejemplo de DataReader, para recuperar varios registros

```
Call conexion()
        cadena = "Select Au_ID, Author  from Authors "
        cmd = New OleDbCommand(cadena, cn)
          dr = cmd.ExecuteReader
              'Do While dr.Read()
              '  ListBox1.Items.Add(dr.Item("Au_ID"))  'se accesa usando el nombre
              '  ListBox2.Items.Add(dr.Item("author"))
              'Loop
              Do While dr.Read()
                  ListBox1.Items.Add(dr.GetInt32(0)) 'se accesa con tipo e indice
                  ListBox2.Items.Add(dr.GetString(1))
              Loop

              MsgBox("error  " & exc.Message)
      dr.Close()
      cn.Close()
```