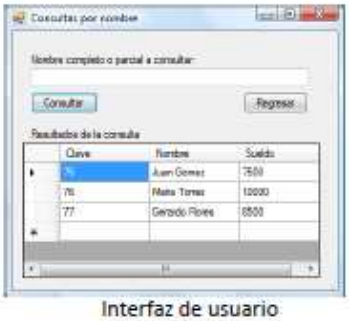
Programación Orientada a Objetos en C#

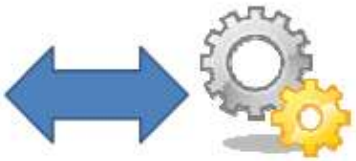
Unidad 6.- Acceso a datos desde C# .NET

Autor:
Dr. Ramón Roque Hernández
http://ramonroque.com/Materias/POOTec.htm
ramonroque@yahoo.com
Colaborador:
Ing. Bruno López Takeyas, M.C.
www.itnuevolaredo.edu.mx/takeyas
takeyas@itnuevolaredo.edu.mx

Arquitectura Programas - Datos



Interfaz de usuario
(Programa de aplicación)



Motor de base de datos

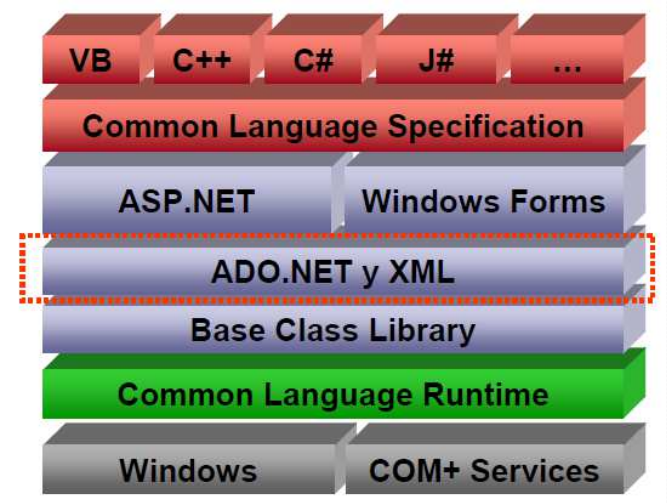
ADO.NET
(ActiveX Data Objects .NET)

- Componente de la plataforma .NET que permite acceder a datos desde un programa
- Es un conjunto de clases, interfaces, estructuras y enumeraciones que permiten trabajar de manera conectada o desconectada con los datos
- ADO.NET puede ser utilizado desde cualquier lenguaje .NET
- ADO.NET es la nueva versión de ADO, creada totalmente a partir de cero.

Las clases de ADO.NET

- ADO.NET es un conjunto de clases pertenecientes al espacio de nombres `System.Data`:
  - `System.Data`
  - `System.Data.Common`
  - `System.Data.OleDB`
  - `System.Data.SqlClient`
- Conjunto de componentes para crear aplicaciones distribuidas de uso compartido de datos.
- Los componentes están diseñados para separar el acceso a los datos de la manipulación de los mismos.

Arquitectura Framework .NET



Evolución histórica

- **ODBC** (Open DataBase Connectivity)
  - Interoperatibilidad con amplio rango de SGBD
  - API acceso ampliamente aceptada
  - Usa SQL como lenguaje de acceso de datos
- **DAO** (Data Access Objects)
  - Interfase de programación para bases de datos JET/ISAM (p. ejem. Access)

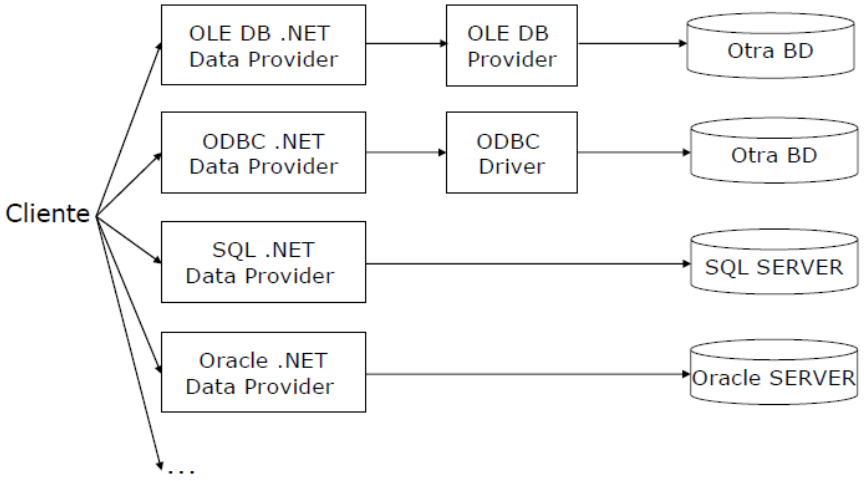
Evolución histórica

- **RDO** (Remote Data Objects)
  - Estrechamente ligado a ODBC
  - Orientada a aplicaciones cliente/servidor
- **OLE DB** (Object Linking and Embedding for Databases)
  - No restringido a acceso de datos relacionales
  - No limitado a SQL como lenguaje de recuperación de datos
  - Tecnología desarrollada por Microsoft
  - Construido sobre COM (Component Object Model)
  - Proporciona una interfase a bajo nivel en C++

Evolución histórica

- **ADO** (ActiveX Data Objects)
  - Ofrece una interfase orientada a objetos
  - Proporciona un modelo de programación para OLE DB accesible desde lenguajes diferentes a C++
  - Diseñado como modelo conectado, altamente acoplado
  - Indicado para arquitecturas cliente/servidor

Proveedores de acceso a datos



Proveedores de acceso a datos

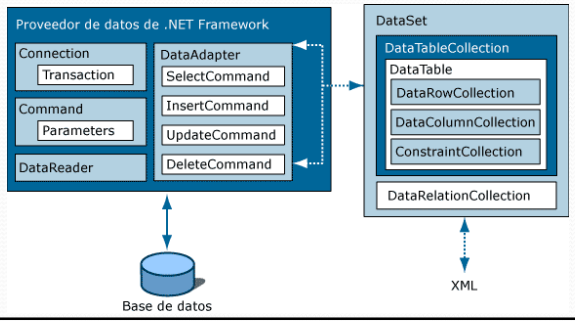
- Conjunto de clases que implementan una serie de interfaces comunes
- **ADO.NET**
  - **OLE DB**
    - Acceso vía protocolo OLE DB a cualquier fuente de datos que lo soporte
    - `System.Data.OleDb`
  - **ODBC**
    - Acceso vía protocolo ODBC a cualquier fuente de datos que lo soporte
    - `System.Data.Odbc`
  - **SQL Server**
    - Acceso nativo a MS SQL Server 7.0 ó superior y MS Access
    - `System.Data.SqlClient`
  - **Oracle**
    - Acceso nativo a Oracle Server
    - `System.Data.OracleClient`
  - **Otros provistos por terceros**
    - MySQL, PostgreeSQL, DB2, etc.

Proveedores de acceso a datos

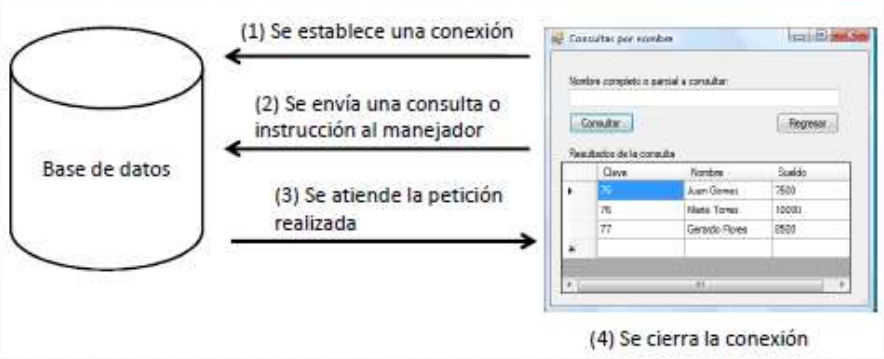
```
public static void ListProviders() {

   DataTable factoryTable = DbProviderFactories.GetFactoryClasses();
   foreach (DataRow dr in factoryTable.Rows) {
        Console.WriteLine("Name: {0}", dr["Name"]);
        Console.WriteLine("Description: {0}", dr["Description"]);
        Console.WriteLine("InvariantName: {0}", dr["InvariantName"]);
        Console.WriteLine("AssemblyQualifiedName: {0}",
                dr["AssemblyQualifiedName"]);
   }
}

// Sample Output

//Name: SqlClient Data Provider
//Description: .Net Framework Data Provider for SqlServer
//InvariantName: System.Data.SqlClient
//AssemblyQualifiedName: System.Data.SqlClient.SqlClientFactory,
//    System.Data, Version=2.0.0.0, Culture=neutral,
//    PublicKeyToken=b77a5c561934e089
```

Componentes de ADO.NET

- **Connection** (conexión)
- **Command** (órdenes)
- **DataReader** (lector de datos)
- **DataAdapter** (adaptador de datos)



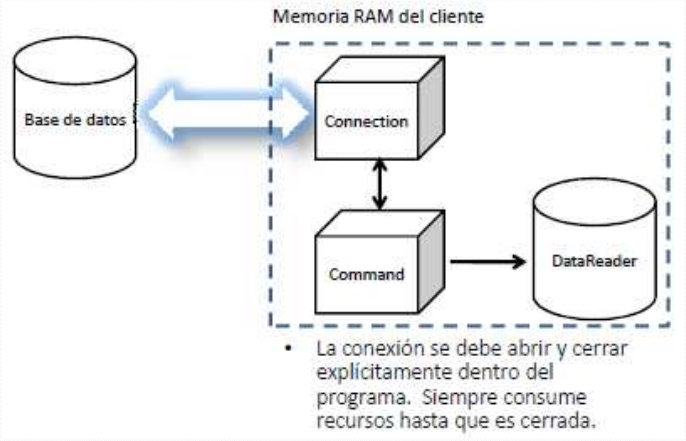
Datos conectados



Objetos del modelo conectado

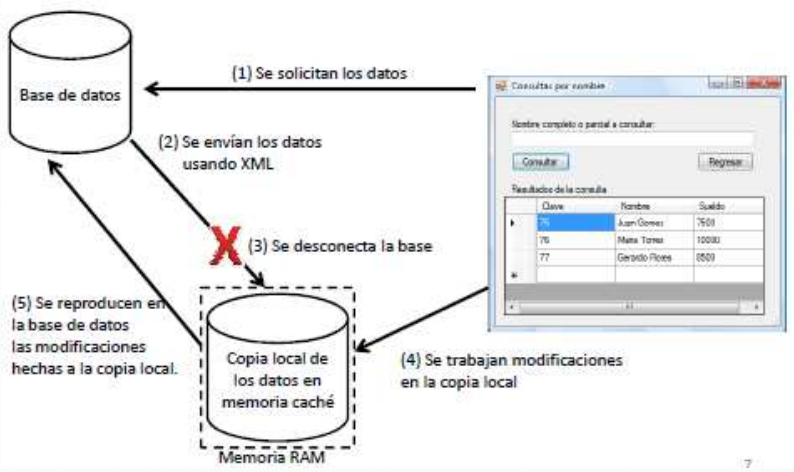
- **Connection** (conexión)
  - Representa una conexión a la BD
  - Permite abrir y cerrar la conexión a la BD
- **Command** (comando)
  - Representa una vía para representar sentencias SQL a la BD
  - Ejemplo: Select, Insert, Delete, Update
- **DataReader** (lector de datos)
  - Almacén temporal de datos, de sólo lectura y sólo hacia adelante

Modelo conectado



- La conexión se debe abrir y cerrar explícitamente dentro del programa. Siempre consume recursos hasta que es cerrada.

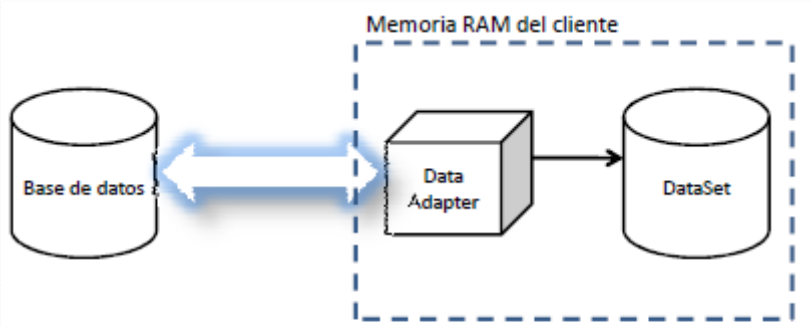
Datos desconectados



Objetos del modelo desconectado

- **DataAdapter** (adaptador de datos)
  - Conecta el programa con la BD, realiza consultas, llena los DataSet y sincroniza los cambios en la BD
  - Es un mediador entre el DataSet y la BD
- **DataSet** (conjunto de datos)
  - Es una estructura para almacenar datos
  - Es una "copia en memoria local" de una porción de la BD
  - Se encuentra en la memoria del cliente
  - Compatible con las BD relacionales (almacena datos en forma de tablas)

Modelo desconectado



- El DataAdapter se conecta a la base de datos, y realiza la consulta, pero NO mantiene una conexión activa con la base de datos.

ADO.NET

- No depende de conexiones continuamente activas, esto es, las aplicaciones se conectan a la BD sólo durante el tiempo necesario para consultar o actualizar datos.
- Las interacciones con la BD se realizan mediante órdenes para acceso a los datos.
- Los datos requeridos normalmente se almacenan en memoria caché en conjunto de datos, lo que permite trabajar sin conexión sobre una copia temporal de los datos.

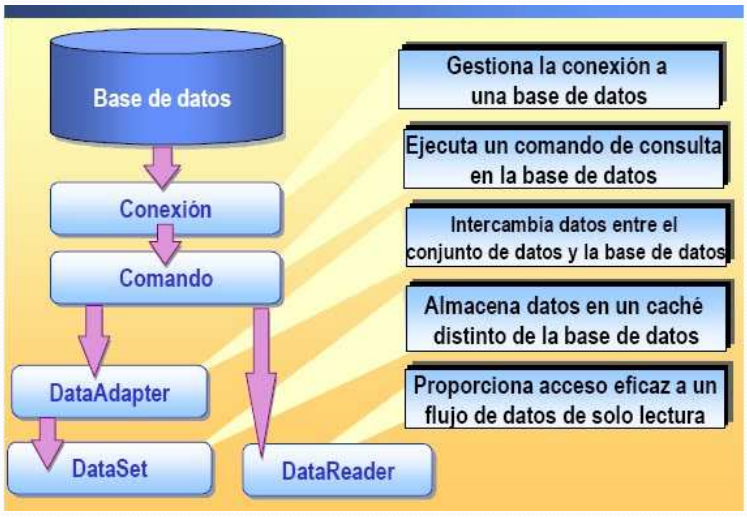
XML

- En ADO.NET, el formato de transferencia es XML.
- La representación de datos XML no utiliza información binaria, sino se basa en texto.
- Muchos servidores bloquean la información binaria.
- Cuando se manejan datos en formato de texto, se pueden enviar mediante cualquier protocolo, como HTTP.

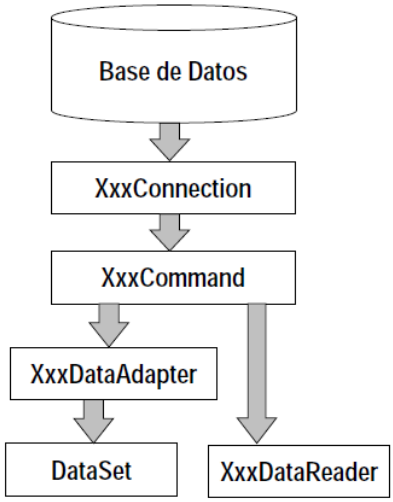
ADO.NET y XML

- ADO.NET está estrechamente integrado con XML

Ejemplo de uso de XML en una aplicación ADO.NET desconectada



Arquitectura



- **XxxConnection:** maneja la conexión a una BD
- **XxxCommand**: ejecuta comandos contra una BD
- **XxxDataAdapter**: intercambia datos entre un DataSet y una BD
- **DataSet**: copia local de datos relacionales (Entorno Desconectado)
- **XxxDataReader**: Proporciona acceso a datos Read-only, Forward-only (Entorno Conectado)

Tareas habituales en la programación de bases de datos

En numerosas aplicaciones, la conexión se cierra después de que el usuario accede a los datos y vuelve a abrirse cuando el usuario transmite actualizaciones o realiza más solicitudes

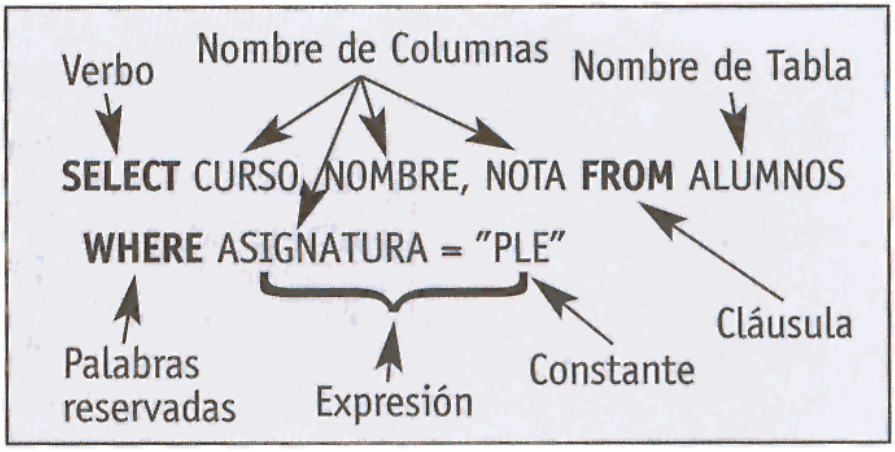
SQL (Structured Query Language)

1. Lenguaje que permite manipular datos mediante el DML (Data Manipulation Language):
   - Obtener datos almacenados en las tablas
   - Insertar, borrar o actualizar datos de las tablas
2. Definir elementos mediante el DDL (Data Definition Language):
   - Crear, borrar, modificar tablas, relaciones, restricciones, etc.

Sentencias de SQL

- Permiten consultar datos de las tablas.
- Típicamente consisten de 3 partes:
  - **SELECT** [Nombres de los campos]
  - **FROM** [Nombre de la(s) tabla(s)]
  - **WHERE** [Condición(es) de filtrado de datos]

Ejemplo de SQL



Definición: SQL es un lenguaje estándar del mercado que ha evolucionado hasta convertirse en el medio de mayor aceptación para realizar consultas y modificar datos de una base de datos

- **Sintaxis de instrucciones SQL habituales**
  - Para especificar exactamente qué registros se desea recuperar, utilizar
    SELECT *Campo* FROM *Tabla*
  - Para limitar la selección de registros, utilizar
    SELECT * FROM *Tabla* WHERE *Campo* = "*String*"
  - Para devolver registros en orden ascendente, utilizar
    SELECT * FROM *Tabla* ORDER BY *Campo* ASC
- **Ejemplo**

```
SELECT Nombre FROM Empleados
```

Videos consultas SQL

- Consultas simples SQL - La sentencia SELECT (http://www.youtube.com/watch?v=IbafcdsR1YA&feature=related)
- Consultas simples SQL - La clausula WHERE (http://www.youtube.com/watch?v=htajN9wZFYk&feature=related)

El objeto Connection

- Para conectarse a una BD, ADO.NET proporciona el objeto **`Connection`**.
- Métodos más usados:
- **`Open()`**.- Abre la conexión. Requiere una cadena de tipo `string` que describa:
  - El tipo de la BD
  - La ubicación
  - Autenticación (si requiere)
- **`Close()`**.- Cierra la conexión previamente abierta.

Connection

Tipo de base de datos	Objeto Connection
SQL Server	`SqlConnection`
OLE DB	`OleDbConnection`
ODBC	`OdbcConnection`
Oracle	`OracleConnection`

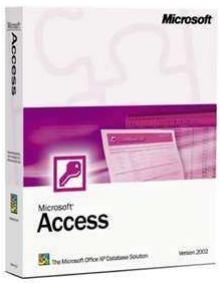
Ejemplo:

```
using System.Data.OleDb;
OledbConnection Conexión = new OleDbConnection(CadenaConexión);
```

Crear una BD en Microsoft Access

- Crear la `BDAlumnos.accdb` en Microsoft Access y la tabla `TablaAlumnos` con los siguientes campos:

Campo	Tipo
Clave	Texto(9)
Nombre	Texto
Semestre	Número (entero)
Promedio	Número (real con 2 decimales)



- Insértele algunos registros

Ejemplo de una conexión con una BD de Microsoft Access

```
using System.Data.OleDb; // Para uso de la base de datos en Access
private OleDbConnection Conexion;  // Declaración de la conexión

// Declaración de la cadena de conexión
string CadenaConexion = 
@"Provider=Microsoft.ACE.OLEDB.12.0;
  Data Source=C:\DATOS\BDAlumnos.accdb";

Conexion = new OleDbConnection(CadenaConexion);
```

El objeto Comand

- Después de establecer la conexión con la BD, se usa el objeto **Command** para ejecutar sentencias SQL y devolver los resultados.

- **OleDbCommand**.- Datos compatibles con OleDb
- **OdbcCommand**.- Datos compatibles con Odbc
- **SqlCommand**.- Datos compatibles con SQL Server
- **OracleCommand**.- Datos compatibles con Oracle

- Ejemplo:

```
OleDbCommand Comando = new
OleDbCommand("SELECT NoCtrl, Nombre FROM
TablaAlumnos WHERE Semestre=8",Conexión);
```

Ejemplo de uso del objeto Command

```
// Declaración de la consulta
string Consulta = "SELECT * FROM TablaAlumnos";

// Declaración del comando de consulta en la conexión con la BD
Comando = new OleDbCommand(Consulta, Conexion);
```

El objeto DataReader

- Se usa solamente para leer datos de una BD.

- **`OleDbDataReader`**.- Datos compatibles con OleDb
- **`SqlDataReader`**.- Datos compatibles con SQL Server

Ejemplo de uso del DataReader

```
// Abrir la BD
Conexion.Open();

// Ejecutar el comando
OleDbDataReader Lector = Comando.ExecuteReader();

while (Lector.Read())
{
  Console.WriteLine(Lector.GetString(0) + " " +
Lector.GetString(1)+" "+Lector.GetValue(2).ToString());
}

// Cerrar la lectura
Lector.Close();
Lector = null;
```

```
class BaseDatosAlumnos
  {
      private OleDbConnection Conexion;  // Declaración de la conexión con la BD
      private OleDbCommand Comando; // Declaración del comando con sentencias SQL
      private OleDbDataReader Lector; // Declaración del lector de datos

      public void LeerDeBaseDeDatos()
      {
          // Declaración de la cadena de conexión
          string CadenaConexion = @"Provider=Microsoft.ACE.OLEDB.12.0;
             Data Source=G:\DATOS\takeyas\Apuntes\POO\Programas\06.- Bases de datos\BD\BDAlumnos.accdb";
          Conexion = new OleDbConnection(CadenaConexion); // Crear la conexión con la BD

          string Consulta = "SELECT * FROM TablaAlumnos";  // Declaración de la consulta
          Comando = new OleDbCommand(Consulta, Conexion);  // Declaración del comando de consulta en la conexión con la BD

          Conexion.Open();  // Abrir la BD
          Lector = Comando.ExecuteReader();  // Ejecutar el comando lector de datos
          while (Lector.Read())
          {
              Console.WriteLine(Lector.GetString(0) + " " + Lector.GetString(1)+" "+Lector.GetValue(2).ToString());
          }
          Lector.Close();  // Cerrar el lector de datos
          Lector = null;
      }

      public void CerrarConexion()
      {
          // Verifica si está activa la lectura
          if (Lector != null)
              Lector.Close();
          // Verifica si está abierta la conexión con la BD
          if (Conexion != null)
              Conexion.Close();
      }
  }
```

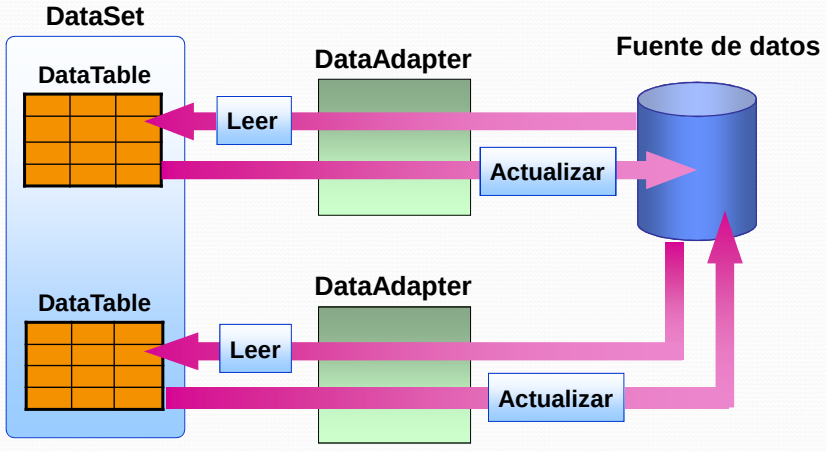
El adaptador de datos

- Conjunto de objetos usado para intercambiar datos entre una BD y un conjunto de datos (DataSet).
- Lee datos de una BD y los coloca en un DataSet para manipularlos.
- Escribe en la BD los datos modificados del DataSet

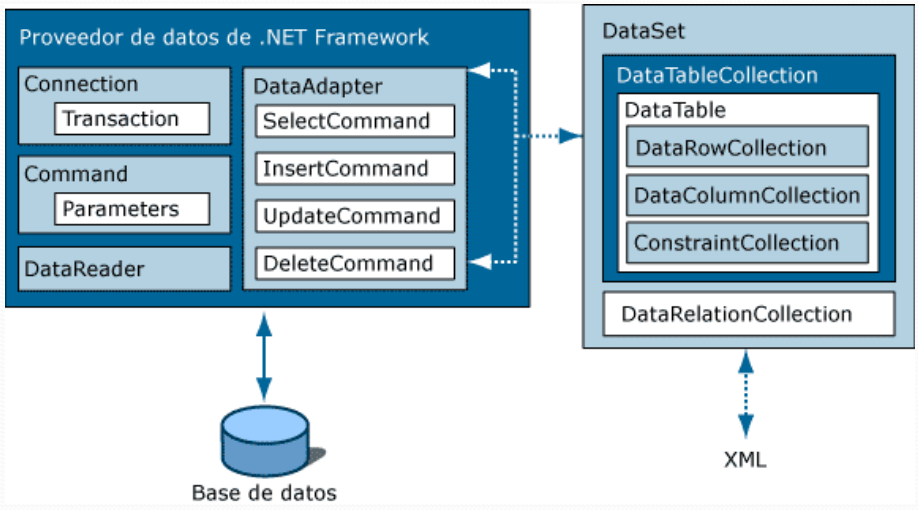
- **`OleDbDataAdapter`**.- Datos compatibles con OleDb
- **`OdbcDataAdapter`**.- Datos compatibles con Odbc
- **`SqlDataAdapter`**.- Datos compatibles con SQL Server
- **`OracleDataAdapter`**.- Datos compatibles con Oracle

- Se crea un adaptador por cada tabla existente en la BD

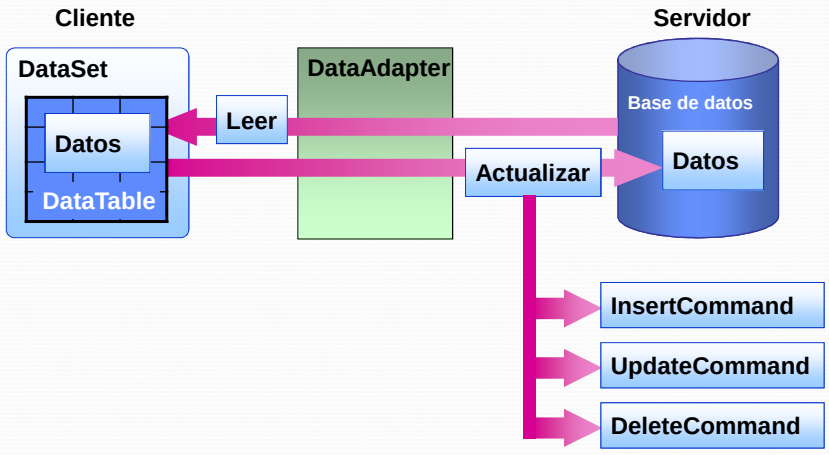
Cómo crear el adaptador de datos



Adaptador de datos



Cómo actualizar una BD en ADO.NET



Cómo crear un registro en la BD

- Crear un nuevo registro que coincida con el esquema de la tabla

```
DataRow miRegistro = dataTable.NewRow();
```

```
miRegistro[“NoCtrl”] = txtNoCtrl.Text;
if(radioButton1.Checked)
    miRegistro[“Sexo”]=radioButton1.Text;
else
    miRegistro[“Sexo”]=radioButton2.Text;
```

- Agregar el nuevo registro al DataSet

```
dataTable.Rows.Add( miRegistro );
```

- Actualizar la base de datos

```
sqlDataAdapter1.Update( dataSet );
```

Cómo borrar un registro de la BD

- Borrar el registro del dataset

```
dataTable.Rows[0].Delete();
```

- Actualizar la BD

```
dataAdapter.Update(dataSet);
```

- Aceptar los cambios en el dataset

```
dataSet.AcceptChanges();
```

Pasos para insertar datos en la BD (modo conectado)

1. Preguntar al usuario los valores que desea insertar.
2. Abrir la conexión a la BD
3. Ejecutar la instrucción SQL correspondiente para inserción de datos:

```
INSERT INTO tabla ( campo1, campo2 … campon)
VALUES (valor1, valor2, valorn )
```

4. Cerrar la conexión
5. Notificar al usuario inserción correcta.

1
```
string nombre = txtNombre.Text;
double sueldo = double.Parse(txtSueldo.Text);
```

2
```
string cc = "Provider=Microsoft.Jet.OLEDB.4.0;" +
                "Data source = c:\\empleados.mdb";
OleDbConnection cn = new OleDbConnection(cc);
cn.Open();
```

3
```
string sql = string.Format("insert into EMPLEADOS (Nombre,Sueldo) " +
                           " values ('{0}' , {1})",nombre, sueldo);
OleDbCommand comando = new OleDbCommand(sql, cn);
int renglones = comando.ExecuteNonQuery();
```

4
```
cn.Close();
```

5
```
if (renglones > 0)
{       MessageBox.Show(" Datos almacenados exitosamente ");
}
else
{       MessageBox.Show(" Los datos NO han sido almacenados ");
}
```

```
static void Agregar()
        {
            // Declaración de la conexión
            OleDbConnection miConexion=null;

            try
            {
                // Declaración y creación de un objeto local
                Alumno miAlumno = new Alumno();

                Console.Clear();
                Console.WriteLine("AGREGAR ALUMNO A LA BASE DE DATOS");

                // Captura de datos del alumno
                Console.Write("\nClave? ");
                miAlumno.Clave = Console.ReadLine();

                Console.Write("Nombre? ");
                miAlumno.Nombre = Console.ReadLine();

                Console.Write("Semestre? ");
                miAlumno.Semestre = int.Parse(Console.ReadLine());

                Console.Write("Promedio? ");
                miAlumno.Promedio = double.Parse(Console.ReadLine());

                // Nombre de la cadena de conexión
                string strCadenaConexion = @"Provider=Microsoft.ACE.OLEDB.12.0; Data Source=C:\DATOS\BDAlumnos.accdb";

                // Establece la conexión a la base de datos
                miConexion = new OleDbConnection(strCadenaConexion);

                // Abre la conexión a la BD
                miConexion.Open();

                // Linea del comando para insertar el alumno
                string strInsertar = "INSERT INTO TablaAlumnos VALUES (@clave, @nombre, @semestre, @promedio)";

                // Establece el comando para insertar los datos del alumno
                OleDbCommand miComando = new OleDbCommand(strInsertar, miConexion);
                miComando.Parameters.AddWithValue("@clave", miAlumno.Clave);
                miComando.Parameters.AddWithValue("@nombre", miAlumno.Nombre);
                miComando.Parameters.AddWithValue("@semestre", miAlumno.Semestre);
                miComando.Parameters.AddWithValue("@promedio", miAlumno.Promedio);

                // Ejecuta el comando
                miComando.ExecuteNonQuery();

                Console.WriteLine("Alumno agregado a la base de datos");
            }
            catch(Exception miExcepcion)
            {
                Console.WriteLine(miExcepcion.Message);
                Console.ReadKey();
            }
            finally
            {
                // Cierra la conexión a la BD
                if (miConexion != null)
                    miConexion.Close();
            }
        }
```

```
static void Reporte()
        {
            Alumno miAlumno = new Alumno();  // Declaración y creación de un objeto local
            OleDbConnection miConexion = null;   // Declaración de la conexión

            // Nombre de la cadena de conexión
            string strCadenaConexion = @"Provider=Microsoft.ACE.OLEDB.12.0; Data Source=C:\DATOS\BDAlumnos.accdb";
            OleDbDataReader Lector=null;   // Declaración del lector de datos
            Console.Clear();
            try
            {
                miConexion = new OleDbConnection(strCadenaConexion); // Crear la conexión con la BD
                string strConsulta = "SELECT * FROM TablaAlumnos";  // Declaración de la consulta
                OleDbCommand miComando = new OleDbCommand(strConsulta, miConexion);  // Declaración del comando de consulta en la conexión con la BD
                miConexion.Open();  // Abrir la conexión a la BD

                Lector = miComando.ExecuteReader();  // Ejecutar el comando lector de datos

                while (Lector.Read())
                {
                    // Obtiene los datos
                    miAlumno.Clave = Lector.GetString(0);
                    miAlumno.Nombre = Lector.GetString(1);
                    miAlumno.Semestre = int.Parse(Lector.GetValue(2).ToString());
                    miAlumno.Promedio = double.Parse(Lector.GetValue(3).ToString());

                    Console.WriteLine(miAlumno.Clave + "\t" + miAlumno.Nombre + "\t" + miAlumno.Semestre + "\t" + miAlumno.Promedio);
                }
            }
            catch(Exception miExcepcion)
            {
                Console.WriteLine("\n"+miExcepcion.Message);
            }
            finally
            {
                if (Lector != null)   // Cierra el lector de datos
                    Lector.Close();

                if (miConexion != null)  // Cierra la conexión a la BD
                    miConexion.Close();

                Console.ReadKey();
            }
        }
```

Videos BD en C#

- Conectar C# y una base de datos tutorial (http://www.youtube.com/watch?v=VPh9906schM)
- Conectando C# con Access (http://www.youtube.com/watch?v=NWHD7dnT1nw&feature=related)
- Conexión Access-Visual Studio C# (http://www.youtube.com/watch?v=ClN9YShhTDE&feature=related)
- C# and Microsoft Access Database - Part 1 (http://www.youtube.com/watch?v=solrzatPmPA&feature=related)
- C# and Microsoft Access Database - Part 2 (http://www.youtube.com/watch?v=cCz8tV7c43Q&feature=relmfu)

Referencias bibliográficas

- http://maravillosomundodelainformatica.blogspot.mx/2010/05/acceso-datos-con-adonet.html
- http://www.codeproject.com/Articles/8477/Using-ADO-NET-for-beginners
- http://www.freewebs.com/freevbdotnet/resources/MSDN%20Training%20-%20ASP.NET%20-%20Module%203_Using%20Microsoft%20ADO.NET%20to%20Access%20Data.pdf
- http://www.softwareresearch.net/fileadmin/src/docs/teaching/WS04/Prod/05.ADO.NET.pdf