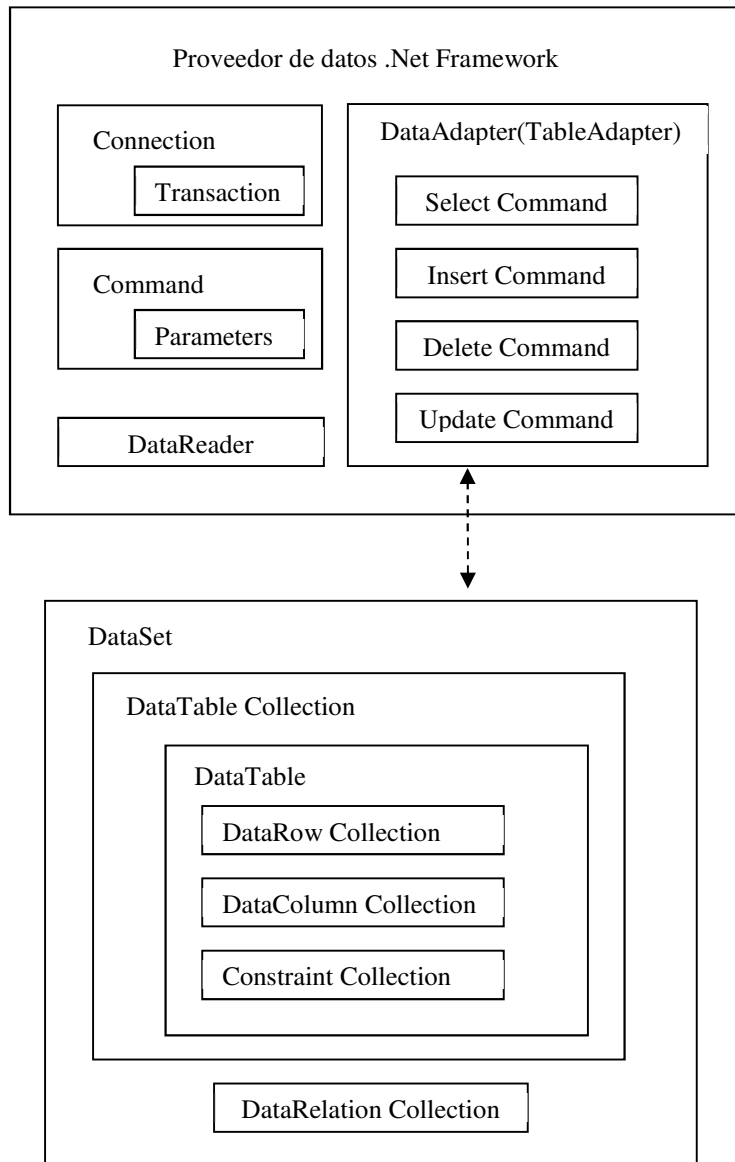


Conexión desde C# a una base de datos en modo cliente servidor

Introducción a la tecnología ADO .Net

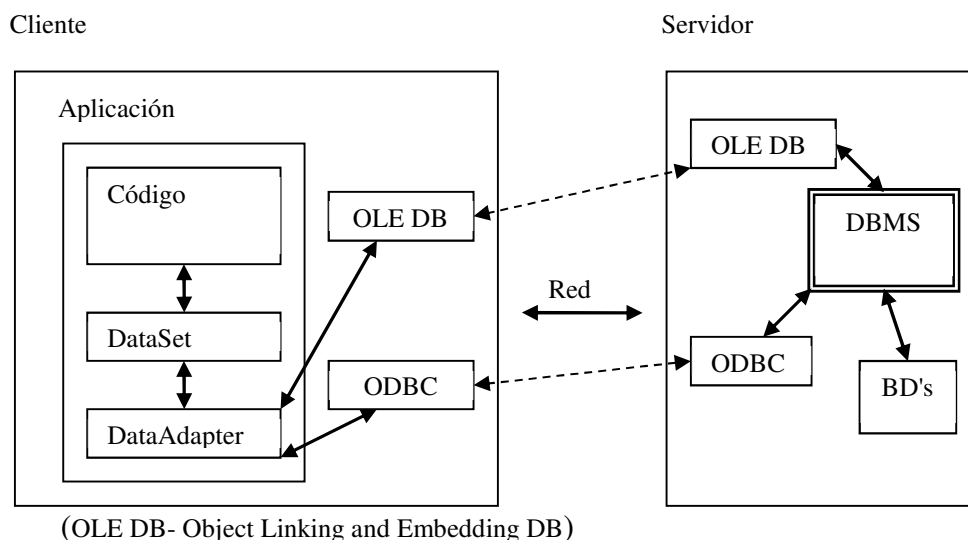
- La tecnología ADO .Net es la que brinda todos los elementos requeridos para acceder y trabajar con una base de datos.
- Consta de dos componentes principales: el **proveedor de datos de .Net Framework** y el **DataSet**:



A continuación se detallan estos componentes:

- **Proveedor de datos de .Net Framework:** es un conjunto de objetos que permite el acceso a la fuente de datos y el intercambio de información entre ésta y la aplicación. Consta de los siguientes objetos:

- **Connection:** usado para conectarse a la base de datos y para el manejo de las transacciones de base de datos. Contiene métodos **open()** y **close()** para abrir y cerrar la conexión a la base.
 - **Command:** para definir las instrucciones SQL a aplicarse a la BD. Tiene una propiedad **Connection** en la cual debe guardarse el objeto de conexión a la BD.
 - **DataReader:** objeto para manejar un cursor read-only, forward-only definido sobre tablas de la base de datos (similar a los cursores básicos de Oracle).
 - **DataAdapter:** sirve de puente entre la fuente de datos y el DataSet o el DataReader. Cuando se hace transferencia de información entre la aplicación y la fuente de datos (en uno u otro sentido), la misma debe pasar a través de este objeto.
- **DataSet:** es un objeto que sirve para almacenar y manipular datos planos, datos XML y datos relacionales. El DataSet reside en memoria central y puede contener tablas, columnas, restricciones, etc., como si se tratará de una base de datos en memoria secundaria. Normalmente se utiliza cuando se quieren combinar y relacionar datos procedentes de varios orígenes o cuando se quieren realizar procesamiento exhaustivos de datos sin necesidad de tener una conexión abierta con la fuente de datos, lo cual la libera para que la utilicen otros clientes.
- Conexión aplicación – BD usando ADO .Net



- 1) La aplicación pide datos a la fuente, por medio del *DataAdapter*, la cual se los envía.
- 2) El *DataAdapter* se desconecta de la fuente de datos (en la mayoría de los casos).
- 3) La aplicación trabaja localmente con los datos (por medio del *DataReader* o del *DataSet*).
- 4) Si los datos se actualizaron (en cuyo caso se trabajó con el *DataSet*, ya que el *DataReader* no permite modificaciones a los datos), la aplicación se reconecta a la fuente de datos y le envía los datos por medio del *DataAdapter*.

Nota: Si se usa ODBC (Open DataBase Connectivity), primero se debe definir el DSN (Data Source Name¹) antes de intentar conectarse a la base de datos.

¹ El DSN se define con: botón Inicio, Panel de Control, Herramientas administrativas, Orígenes de datos (ODBC), pestaña DSN de usuario, <Agregar...>, elegir el controlador: Microsoft Access Driver (para Access) u Oracle en Oracle_Home (para Oracle), <Finalizar>, dar el nombre del DSN, dar la ruta de la BD con <Seleccionar...> (para Access) o el TNS: oracle u oraculo.itam.mx (para Oracle), <Aceptar>.

Organización del DataSet²

- El DataSet se puede ver como una colección de tablas; en el extremo, puede ser una imagen de una base de datos completa, aunque como reside en memoria central, normalmente en el DataSet sólo se tiene una parte de la misma. El DataSet se desconecta de la base de datos en cuanto los datos se cargan en él; si hay cambios posteriores, se reconecta para actualizar las tablas correspondientes en la base de datos.

NOTA: se debe incluir la siguiente declaración en el programa para poder usar estos objetos: `using System.Data;`

- Para usar objetos tipo DataSet se deben declarar variables de ese tipo, por ejemplo:

```
Dataset DSGeneral;
```

- Para acceder a las tablas del DataSet se pueden usar variables de tipo **DataTable**, declaradas como en:

```
DataTable tabla;
```

- A su vez, una tabla se puede ver como una colección de filas (tuplas) las cuales se pueden acceder con variables del tipo **DataRow**, declaradas como en:

```
DataRow fila;
```

- De igual modo, una fila se puede ver como una colección de columnas (atributos) las cuales se pueden acceder con variables del tipo **DataColumn**, declaradas como en:

```
DataColumn columna;
```

- Una vez declaradas las variables, el acceso a una tabla, fila o columna específica se puede hacer (respectivamente) con:

```
tabla= dataset.Tables[indiceTabla] o  
tabla= dataset.Tables[nombreTabla],  
fila= tabla.Rows[indiceFila] o  
foreach (DataRow fila in tabla.Rows),  
columna= fila[indiceColumna] o columna= fila[nombreColumna]
```

- En todos los casos que aplica, el índice inicia a partir de **cero**.

² Antes de iniciar la conexión y si no se hizo cuando se empezó a utilizar el SQL Developer: ir al servidor y copiar desde Docs\InstalConfig a los archivos: sqlnet.ora y tnsnames.ora, después pegarlos en la carpeta C:\App\Oracle\product\12.1.0\client_1\network\admin

Métodos y propiedades importantes de clases de ADO.NET

- **DataAdapter:**
 - **Fill(...)**- Agrega o actualiza filas en el DataSet especificado para que coincidan con las del origen de datos.
 - **Update(DataSet)** o **Update(DataSet.Tabla)**- Actualiza la fuente de datos con todos los cambios realizados en el DataSet. Si se usa la segunda opción, sólo se actualiza la "Tabla" indicada, perteneciente al DataSet.
 - **DeleteCommand**- Obtiene o establece una instrucción Delete o un procedimiento almacenado para eliminar registros de un DataSet.
 - **InsertCommand, UpdateCommand, SelectCommand**- Ídem a la anterior, sólo que para insertar, cambiar o seleccionar registros, respectivamente.

Las propiedades anteriores (que en realidad son objetos) tienen a su vez, las siguientes propiedades y métodos importantes:

- **CommandText**- para obtener o establecer el texto de la instrucción.
 - **CommandType**- ídem para el tipo de instrucción (procedimiento almacenado, tabla o texto).
 - **Parameters**- colección para almacenar y manipular los parámetros. Tiene un método **Add(Objeto)** para agregar parámetros a una instrucción SQL, y un método **Clear()** para limpiar la colección.
- **OleDb:** es una clase que contiene una serie de constructores para crear objetos de ADO.NET. Se puede usar para crear objetos requeridos en el procesamiento de una fuente de datos. Los principales constructores son:
 - **OleDbCommand**- para crear un objeto **Command**.
 - **OleDbConnection**- para crear un objeto **Connection**.
 - **OleDbDataAdapter**- para crear un objeto **DataAdapter**.
 - **OleDbDataReader**- para crear un objeto **DataReader**.
 - **OleDbParameter**- para crear un objeto **Parameter**. Se puede usar para crear parámetros que requiere la ejecución de una instrucción SQL o un procedimiento almacenado.
 - **OleDbTransaction**- para crear un objeto **Transaction**.

NOTA: se debe incluir la siguiente declaración en el programa para poder usar estos objetos: `using System.Data.OleDb;`
 - **OracleClient:** es una clase similar a la anterior, sólo que para trabajar con una base de datos de Oracle (hay que bajarla del sitio de Oracle para poder usarla).
 - **SqlClient:** es una clase similar a la anterior, sólo que para trabajar con una base de datos de SQL Server.
 - **DataSet**
 - **AcceptChanges()**- el DataSet va llevando el registro de todas las modificaciones que se han hecho en los datos que almacena. Al llamar a este método, actualiza completamente el DataSet, quedando registrados los datos tal y como están hasta el momento, perdiendo el rastro de todas las modificaciones que se les hayan hecho previamente.
 - **Clear()**: para eliminar el contenido actual del DataSet.
 - **Tables[...]**- para acceder a una tabla del DataSet (ver: Organización del DataSet).

Visual C# (varios)

1) Ejecución de procedimientos almacenados, de Oracle, desde una aplicación de Visual C#

- a. Hacer la conexión a la base de datos.
- b. Especificar el llamado al procedimiento.
- c. Especificar los parámetros de entrada del procedimiento.
- d. Especificar los parámetros de salida.
- e. Ejecutar el procedimiento (dentro de un try-catch).
- f. Recuperar el (los) valor(es) regresado(s) por el procedimiento.
- g. Cerrar la conexión a la base de datos.

2) Ejecución de funciones almacenadas, de Oracle, desde una aplicación de Visual C#

- a. Hacer la conexión a la base de datos.
- b. Especificar el llamado a la función.
- c. Especificar (con OleDbParameter) lo siguiente: la cadena: "RETURN_VALUE" (como si fuera el nombre de un parámetro de salida); el tipo de parámetro: ParameterDirection.ReturnValue; los demás valores requeridos.
- d. Especificar los parámetros de entrada de la función.
- e. Ejecutar la función (dentro de un try-catch).
- f. Recuperar el valor regresado por la función: con Parameters["RETURN_VALUE"].
- g. Cerrar la conexión a la base de datos.

Nota: si una función en Oracle regresa un valor con decimales, hay problemas para recuperarlo desde Visual C#, ya que no hay un tipo de datos que recupere bien lo que regresa Oracle.

Solución: re-definir la función como un procedimiento para que regrese a través de un parámetro de salida, tipo Number (o Real), el resultado que regresa la función.

3) Biblioteca de componentes

1. Componentes no visuales

- Son otro de los tipos de componentes que ofrece Microsoft dentro de su tecnología COM. Tienen la misma filosofía que los componentes visuales de Visual C#, salvo que no tienen una interfaz visual. Asimismo pueden ser usados en cualquier aplicación.
- Los componentes se guardan también como una biblioteca de enlace dinámico (.dll). El componente puede estar compuesto por **uno o más módulos de clase** en un proyecto Visual C# .Net.

2. Creación del componente no visual

- Creación del proyecto: seleccionar un proyecto nuevo **Biblioteca de clases (Class Library)**, darle nombre: (p. ej.: *GestorBD*), eliminar *clase1.cs*.
- Agregación de las clases del componente, por cada clase: menú breve del *Proyecto*, *Agregar, Componente...*, seleccionar plantilla Clase de componentes (Component Class), darle nombre (p. ej.: *GestorBD*, sin alterar el .cs), <Agregar>, en la ventana que se abre dar clic en: haga clic aquí para cambiar a la vista Código.
- Declaración de propiedades, métodos y eventos de cada clase del componente.
 - Se declaran como en cualquier clase de Visual C#.
- Compilación del componente: una vez que se han construido las clases, hay que compilarlos: menú *Compilar, Generar componente*. Guardar el proyecto (sin crear nuevo directorio). Esto genera el código correspondiente en un archivo *componente.dll* en el directorio bin/ Debug (o bin/ Release) del proyecto.

3. Prueba del componente no visual

- Se puede tener un proyecto de prueba que quede en la misma solución del componente o en una solución separada (nueva). Hay que recordar que el proyecto que usa a la biblioteca, se debe regenerar cada vez que se modifica y recompila el componente.
- Sea en un proyecto en la misma solución o en un proyecto separado, se debe agregar la referencia al .dll: seleccionar el proyecto de prueba, menú breve, *Agregar referencia...*, pestaña *Examinar*, <Examinar...>, seleccionar el archivo .dll (que debe estar en el directorio bin/Debug (o bin/ Release) de la solución del componente), <Aceptar>.
- En el proyecto de prueba: declarar variable(s) –de preferencia con WithEvents para poder atrapar los eventos generados desde el componente– y crear objeto(s) de la(s) clase(s) del componente, programar, ejecutar.

4) Cadenas de conexión a bases de datos según el DBMS (empleando a GestorBD)

Oracle

```
GestorBD = New GestorBD.GestorBD("MSDAORA", "Usuario_BD",  
    "Password_BD", "oracle")
```

SQL Server

```
GestorBD = New GestorBD.GestorBD("SQLNCLI11", "localhost",  
    "Usuario_BD", "Password_BD", "NombreBD")
```

Donde:

Usuario_BD (Oracle y SQL Server): es la cuenta de la base de datos.

Password_BD (Oracle y SQL Server): es la contraseña respectiva.

NombreBD (SQL Server): es el nombre de la base de datos.

Nota: si la conexión a Oracle no se puede hacer, cambiar el primer parámetro del constructor por: OraOLEDB.Oracle.1

Entity Framework

Es otra de las tecnologías de Microsoft .NET para acceder a una base de datos relacional. Su principal característica es que hace la transformación relacional-objetos de una manera casi transparente para el desarrollador, y un poco más fácil que en el caso de los DataSets.

Microsoft sugiere usar Entity Framework (EF) para las aplicaciones nuevas que se hagan con Visual Studio (VS) y Bases de Datos (BD). Sugiere utilizar los datasets, una tecnología más sencilla y menos costosa que EF, para desarrollo y pruebas rápidas, y para procesamiento de datos de una complejidad, o volumen, bajos o medianos.

Proceso para crear una aplicación Visual C# - Base de Datos

1. Incorporación de EF a Visual Studio

- a. Verificar si EF está incorporado al IDE de Visual Studio: ir al Explorador de soluciones del proyecto, colocarse en el proyecto, expandir Referencias, verificar si el Entity Framework está en las bibliotecas que aparecen ahí. **Si ya está, se termina** este punto; **si no**, pasar al siguiente inciso.
- b. Agregar EF al IDE: ir al Explorador de soluciones y seleccionar al proyecto, menú breve³, elegir: Administrar paquetes NuGet, en el menú Examinar: buscar e instalar el paquete Entity Framework (normalmente es el primero). Al terminar la instalación, ya debe aparecer en la entrada References mencionada en el inciso anterior.

2. Crear el modelo de la base de datos

Para acceder a una BD usando EF existen cuatro maneras de hacerlo: dos con código y dos con un diseñador de VS; de estas, hay para los casos de que la BD exista o no. Aquí se describe el caso de usar el **diseñador** y que la **BD** ya se haya **creado** previamente (el ejemplo es con una BD de SQL Server: SistemaEscolar). Se genera un modelo de la BD que es el usado para interactuar con ésta desde el programa Visual C#.

- a. En el Explorador de soluciones: elegir el proyecto, menú breve, Agregar, Nuevo elemento.
- b. En la ventana Agregar nuevo elemento: elegir Datos, ADO.NET Entity Data Model, darle un nombre al modelo (p. ej., SistemaEscolar_Modelo), <Agregar>.
- c. En la ventana Asistente para Entity Data Model: elegir EF Designer desde base de datos, <Siguiente>, <Nueva conexión>.
 - En la ventana Propiedades de la conexión: Origen de datos: (valor default), Nombre del servidor: localhost o el nombre de la computadora, Autenticación: Autenticación de SQL Server, dar el nombre de usuario y la contraseña, marcar: Guardar mi contraseña, Seleccionar el nombre de la BD: SistemaEscolar, <Probar conexión>, debe salir un mensaje de éxito, <Aceptar>.
- d. Marcar el botón de radio: Sí, <Siguiente>, <Siguiente>, marcar todas las tablas de la BD, <Finalizar>. Con esto ya se tiene el modelo de la BD dentro de la aplicación.

³ Menú breve- se refiere al botón derecho del ratón.

- e. Antes de usar este modelo en la programación, se requiere hacer los siguientes cambios en el archivo: SistemaEscolar_Modelo.tt:
- Reemplazar las dos ocurrencias de ICollection con ObservableCollection.
 - Reemplazar la primera ocurrencia de HashSet con ObservableCollection (alrededor de la línea 51). No reemplazar la segunda ocurrencia de HashSet.
 - Reemplazar la única ocurrencia de System.Collections.Generic con System.Collections.ObjectModel (alrededor de la línea 431).
- f. Después de esto, ya se puede utilizar el modelo en la programación. Cuidar el agregar en cada forma donde se use el modelo: `using System.Data.Entity;`