Introducción a ADO .NET

ADO.NET forma parte del tercer nivel del conjunto de objetos que el **.NET Framework** ofrece para trabajar dentro de esta plataforma, junto con XML constituyen un subgrupo específico que están preparados para manejar datos. Al igual que el resto de los elementos que constituyen el .**NET Framework** se estarán ejecutando dentro del **Motor de Ejecución Común** o **CLR** y está formado por un conjunto de clases administradas organizadas en espacios de nombre.

ADO.NET proporciona acceso coherente a orígenes de datos como Microsoft SQL Server, así como a orígenes de datos expuestos mediante OLE DB y XML (Microsoft Access, Microsoft Visual FoxPro, etc.). Las aplicaciones para usuarios que comparten datos pueden utilizar ADO.NET para conectarse a estos orígenes de datos y recuperar, manipular y actualizar los datos.

ADO.NET separa limpiamente el acceso a datos de la manipulación de datos y crea componentes discretos que se pueden usar por separado o conjuntamente. ADO.NET incluye proveedores de datos de .NET Framework para conectarse a una base de datos específica, ejecutar comandos y recuperar resultados. Esos resultados se procesan directamente o se colocan en un objeto **DataSet** (repositorio en cliente) de ADO.NET con el fin de exponerlos al usuario para un propósito específico, junto con datos de varios orígenes, o de utilizarlos de forma remota entre niveles. El objeto **DataSet** de ADO.NET también puede utilizarse independientemente de un proveedor de datos de .NET Framework para administrar datos que son locales de la aplicación o que proceden de un origen XML.

ADO.NET fue diseñado para tener un acceso desconectado a la base de datos. En principio en ADO.NET se puede modelar la estructura de una base de datos desde el código sin necesidad de acceder a una base de datos existente.

Proveedores de Datos .NET

Los proveedores permiten que una aplicación lea y escriba los datos almacenados en una fuente de datos. ADO.NET soporta tres proveedores principales:

Proveedor	Descripción
OLE DB .NET	Permite acceder a una fuente de datos para la que exista un proveedor OLE DB, aunque a expensas de un conmutador para administrar código no administrado, con la consiguiente degradación de rendimiento.
SQL Server .NET	Ha sido escrito específicamente para acceder a SQL Server 7.0 o versiones posteriores, utilizando Tabular Data Stream (TDS) como medio de comunicación. TDS es el protocolo nativo de SQL Server, por lo que se puede confiar en que este proveedor ofrezca mejores prestaciones que el proveedor de datos OLE DB.
ODBC .NET	Este proveedor funciona como un puente a una fuente ODBC, por lo que en teoría se puede utilizar para acceder a cualquier fuente para la que exista un controlador ODBC.

Un proveedor de datos de .NET Framework sirve para conectarse a una base de datos, ejecutar comandos y recuperar resultados. Esos resultados se procesan directamente o se colocan en un **DataSet** de ADO.NET con el fin de exponerlos al usuario para un propósito específico, junto con datos de varios orígenes, o de utilizarlos de forma remota entre niveles. El diseño del proveedor de datos de .NET Framework hace que sea ligero, de manera que cree un nivel mínimo entre el origen de datos y su código, con lo que aumenta el rendimiento sin sacrificar la funcionalidad.

En la tabla siguiente se describen los cuatro objetos centrales que constituyen un proveedor de datos de .NET Framework.

Objeto	Descripción
Connection	Establece una conexión a un origen de datos determinado.
Command	Ejecuta un comando en un origen de datos. Expone una colección de parámetros (**Parameters**) y puede ejecutarse en el ámbito de un objeto de transacción (**Transaction**) de **Connection**.
DataReader	Lee una secuencia de datos de un origen de datos. Es de sólo avance y sólo lectura.
DataAdapter	Llena un **DataSet** y realiza las actualizaciones necesarias en el origen de datos subyacente.

Proveedor de datos de .NET Framework para OLE DB

El proveedor de datos de .NET Framework para OLE DB utiliza OLE DB nativo para permitir el acceso a datos mediante la interoperabilidad COM. El proveedor de datos de .NET Framework para OLE DB admite tanto transacciones locales como las transacciones distribuidas.

Las clases del proveedor de datos de .NET Framework para OLE DB están ubicadas en el espacio de nombres **System.Data.OleDb**.

A continuación se muestran algunos de los proveedores que se han probado con ADO.NET:

Controlador	Proveedor
SQLOLEDB	Proveedor OLE DB para SQL Server de Microsoft.
MSDAORA	Proveedor OLE DB para ORACLE de Microsoft.
Microsoft.Jet.OLEDB.4.0	Proveedor OLE DB para Microsoft Jet.

Proveedor de datos de .NET Framework para SQL Server

Es ligero y presenta un buen rendimiento porque está optimizado para tener acceso a SQL Server directamente, sin agregar una capa OLE DB u ODBC intermedia. En la siguiente ilustración se compara el proveedor de datos de .NET Framework para SQL Server y el proveedor de datos de .NET Framework para OLE DB.

Para utilizar el proveedor de datos de .NET Framework para SQL Server, debe tener acceso a Microsoft SQL Server 7.0 o posterior. Las clases del proveedor de datos de .NET Framework para SQL Server están ubicadas en el espacio de nombres **System.Data.SqlClient**. Para las versiones anteriores de Microsoft SQL Server, use el proveedor de datos de .NET Framework para OLE DB con el proveedor OLE DB de SQL Server (SQLOLEDB).

Proveedor de datos de .NET Framework para ODBC

El proveedor de datos de .NET Framework para ODBC utiliza el Administrador de controladores ODBC nativos para permitir el acceso a datos mediante la interoperabilidad COM. El proveedor de datos de ODBC admite tanto transacciones locales como transacciones distribuidas.

En la siguiente tabla se muestran los controladores ODBC que se han probado con ADO.NET:

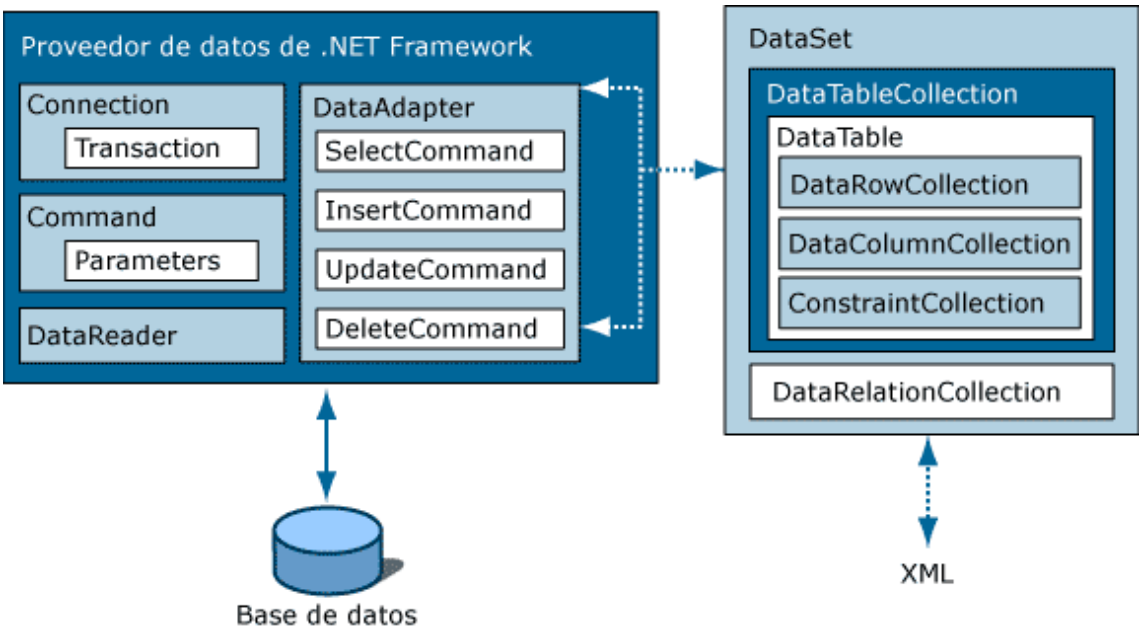
Controlador
Microsoft SQL Server
Microsoft ODBC para ORACLE
Microsoft Access Driver (*.mdb)

Las clases del proveedor de datos de .NET Framework para ODBC están ubicadas en el espacio de nombres **System.Data.Odbc**.

Modelo de Objetos ADO .NET

El objeto **Connection** proporciona conectividad con un origen de datos. El objeto **Command** permite tener acceso a comandos de base de datos para devolver datos, modificar datos, ejecutar procedimientos almacenados y enviar o recuperar información sobre parámetros. El objeto **DataReader** proporciona una secuencia de datos de alto rendimiento desde el origen de datos. Por último, el objeto **DataAdapter** proporciona el puente entre el objeto **DataSet** y el origen de datos. El **DataAdapter** utiliza objetos **Command** para ejecutar comandos SQL en el origen de datos tanto para cargar el **DataSet** con datos como para reconciliar en el origen de datos los cambios aplicados a los datos incluidos en el **DataSet**.
Es posible escribir proveedores de datos de .NET Framework para cualquier origen de datos. .NET Framework incluye dos proveedores de datos de .NET Framework: el proveedor de datos de .NET Framework para OLE DB y el proveedor de datos de .NET Framework para SQL Server.

Arquitectura de ADO .NET



Espacio de Nombres de ADO .NET

(Espacios de nombre principales)

Espacio de Nombre	Descripción
System.Data	Reúne los objetos de **ADO.NET** que no pertenecen a un proveedor de datos específico, además incluye varias interfaces de ADO.NET genéricas.
System.Data.Common	Contiene los objetos **DataAdapter** y otras clases virtuales. Estas clases se utilizan como clases bases para varios objetos en los espacios de nombres siguientes.
System.Data.OleDb	Contiene objetos asociados al proveedor de datos OLE DB .NET, tales como **OleDbConnection**, **OleDbCommand**, **OleDbDataReader** y **OleDbDataAdapter**.
System.Data.SqlClient	Contiene los objetos asociados al Proveedor de datos SQL Server .NET, tales como **SqlConnection**, **SqlCommand**, **SQLDataReader** y **SQLDataAdapter**.

ADO .NET en Modo Conectado

ADO .NET es un tema muy amplio por lo que se ha decidido separarlo en dos capítulos. En esta primera parte se va a utilizar un escenario de Modo Conectado, es decir manteniendo abierta la conexión de la Base de Datos.

El Objeto Connection

Si se trabaja en modo conectado tanto como sin conexión, lo primero que se debe hacer cuando se trabaja con una fuente de datos es abrir una conexión a ella. En ADO .NET esto implica que se crea un objeto **Connection** que conecta con la base de datos específica.

```
'Creando la Cadena de Conexión
String  CadConexion = “Provider=Microsoft.ACE.OLEDB.12.0;Data
Source=C:\myFolder\myAccess2007file.accdb;”;
OleDbConnection Conexion = New OleDbConnection(CadConexion) ;
'Abriendo la Conexión
Conexion.Open() ;
'Cerrando la Conexión
Conexion.Close();
```

El Objeto Command

Una vez que se abre una conexión, se puede elegir entre trabajar en modo conectado o sin conexión. En el primer caso, debe crearse un objeto **Command** que contenga una consulta de selección o una consulta de acción y a continuación ejecutar uno de los métodos **Execute***xxx*, en el que el nombre exacto depende del tipo de consulta.

Método	Descripción
ExecuteNonQuery	Ejecuta la consulta de acción especificada en **CommandText**, y devuelve el número de filas afectadas.
ExecuteReader	Ejecuta la consulta de acción especificada en **CommandText**, devuelve el objeto **DataReader**
ExecuteScalar	Ejecuta la consulta de acción especificada en **CommandText**, devuelve el valor escalar de la primera columna de la primera fila, ignorando el resto de valores.

ExecuteNonQuery

```
OleDbConnection  Conexion ;
OleDbCommand Comando;
int NroFilasAfectadas;
Try {
        Conexion = New OleDbConnection("Provider=Microsoft.ACE.OLEDB.12.0;Data
        Source=C:\myFolder\myAccess2007file.accdb;") ;
        Conexion.Open() ;
        Comando = New OleDbCommand("DELETE FROM ALUMNO", Conexion) ;
        NroFilasAfectadas = Comando.ExecuteNonQuery() ;
        Return (NroFilasAfectadas.ToString & " filas eliminadas") ;
}Catch( Exception ex){
        Throw new exception(ex.Message) ;
}
```

ExecuteReader

```
OleDbConnection Conexion;
OleDbCommand  Comando;
OleDbDataReader Lector;
Try {
        Conexion = New OleDbConnection("Provider=Microsoft.ACE.OLEDB.12.0;Data
        Source=C:\myFolder\myAccess2007file.accdb;");

        Conexion.Open() ;
        Comando = New OleDbCommand("SELECT * FROM ALUMNO", Conexion) ;
        Lector = Comando.ExecuteReader(CommandBehavior.CloseConnection) ;
        While( Lector.Read ){
                ListBox1.Items.Add(Lector("Apellidos")) ;
        }
        Lector.Close() ;
}Catch(Exception ex){
        Throw new exception(ex.Message) ;
}
```

El método tiene un parámetro opcional codificado por bit, **CommandBehavior**, los valores disponibles son:

Valor de enumerado	Descripción
CloseConnection	La conexión se debe cerrar inmediatamente después que objeto **DataReader** se cierre.
SingleRow	Se espera que la instrucción SQL devuelva una sola fila de datos.
SingleResult	Se espera que la instrucción SQL devuelva un solo valor escalar.
KeyInfo	Las consultas de información de columnas y de la clave principal se ejecutan sin cerrar las filas seleccionadas.
SequencialAccess	Los resultados de la consulta se leen secuencialmente en el nivel de columnas en lugar de ser devueltos como un bloque completo.
SchemaOnly	La consulta sólo devuelve información de columnas y no afecta al estado de la base de datos.

ExecuteScalar

Utiliza el método **ExecuteScalar** para recuperar un valor único desde una base de datos. Esto requiere menos código que utilizar el método **ExecuteReader** y luego realizar las operaciones necesarias para generar el valor único utilizando los datos devueltos por un **OleDbDataReader**.

```
OleDbConnection Conexion;
OleDbCommand  Comando;
OleDbDataReader Lector;
Try {
        Conexion = New OleDbConnection("Provider=Microsoft.ACE.OLEDB.12.0;Data
        Source=C:\myFolder\myAccess2007file.accdb;");

        Conexion.Open() ;
        Comando = New OleDbCommand ("SELECT COUNT(*) As Cantidad FROM ALUMNO", Conexion) ;
        NroAlumnosExistentes = (Int) Comando.ExecuteScalar();
        Return (NroAlumnosExistentes.ToString & " alumnos registrados")
}Catch(Exception ex){
        Throw new exception(ex.Message) ;
}
```

El Objeto DataReader

Cuando se recupera una gran cantidad de datos, mantener abierta la conexión se vuelve un problema. Para resolver esto, el **DataReader** es un flujo de sólo hacia delante y sólo lectura devuelto desde la base de datos. En la memoria se mantiene sólo un registro a la vez. Debido a que su funcionalidad es muy especifica (o limitada), es *"ligero"*. Esto es especialmente cierto si lo compara utilizando **DataReader** con utilizar **DataSet**.

Un **DataReader** tiene diversas formas de devolver los resultados. Analice los siguientes ejemplos que muestran cómo se puede acceder a las columnas de cada fila obtenida desde la base de datos:

Accediendo a un campo mediante su nombre y la propiedad **Item**

```
ListBox1.Items.Add(Lector.Item("Apellidos"))
```

Accediendo a un campo mediante su nombre (la propiedad **Item** está marcada como **Default**)

```
ListBox1.Items.Add(Lector("Apellidos"))
```

Accediendo a un campo mediante su posición en la tabla. Devuelve un **Object**.

```
ListBox1.Items.Add(Lector(1).ToString)
```

Accediendo a un campo mediante su posición y la función **GetString**. Si sabe qué tipo de dato contiene la columna puede evitar conversiones utilizando directamente la función apropiada según el tipo de dato de la columna.

```
ListBox1.Items.Add(Lector.GetString(1))
```