**ADO: Activex Data Objects**

El presente introduce conceptos básicos de ADO.

ADO es uno de los mecanismos que usan los programas de computadoras para comunicarse con las bases de datos, darles órdenes y obtener resultados de ellas.

Con ADO, un programa puede leer, insertar, editar, o borrar, la información contenida en diferentes áreas de almacenamiento dentro de la base de datos llamadas tablas. Además, se puede manipular la propia base de datos para crear nuevas áreas para el almacenamiento de información (tablas), como también alterar o eliminar las ya existentes, entre otras cosas. Fue desarrollado por Microsoft y es usado en ambientes Windows por lenguajes de programación como Visual Basic, C++, C# entre otros, como también en la Web mediante el uso de Active Server Pages (ASP) y el lenguaje VBScript.

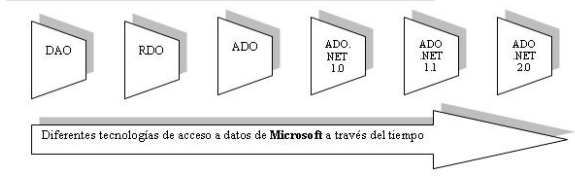
Evolución

ADO substituyó tanto a DAO (Data Access Object), como a RDO (Remote Data Object), que eran los sistemas previos que se usaban para acceder a las bases de datos y bases de datos remotas, respectivamente. Tiene la mayor parte de la funcionalidad de ambos modelos y sin embargo es más sencillo de usar y de entender y por lo tanto más fácil y menos engorroso de programar. La última versión de ADO, creada por Microsoft, se llama ADO.NET, y se usa en los entornos de programación de la plataforma .NET, de Microsoft, para manejar bases de datos tanto en Windows como en la Web mediante ASP.NET, que es la nueva versión del ASP para la plataforma.NET.

Visual Studio

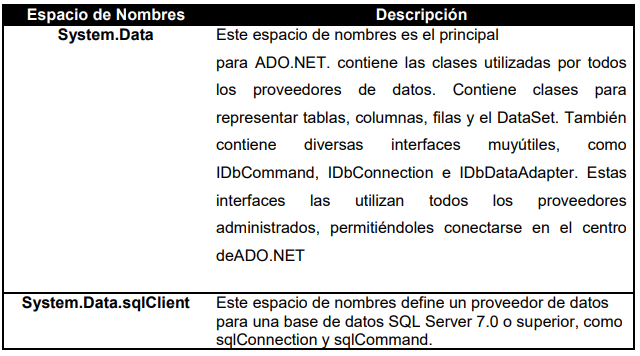
En Visual Basic, tres interfaces de acceso de datos están disponibles para usted: ActiveX Data Objects (ADO), objetos de datos remotos (RDO), y Data Access Objects (DAO). Una interfaz de acceso a datos es un modelo de objetos que representa varias facetas del acceso a los datos. Con Visual Basic, puede controlar mediante programación la conexión, los constructores de sentencias y los datos devueltos para usar en cualquier aplicación.

¿Por qué hay tres interfaces de acceso a datos en Visual Basic? La tecnología de acceso a datos evoluciona constantemente, y cada una de las tres interfaces representa un estado del arte diferente. La última es ADO, que presenta un modelo de objetos más simple, aunque más flexible, que RDO o DAO. Para nuevos proyectos, debe usa ADO como su interfaz de acceso a datos.

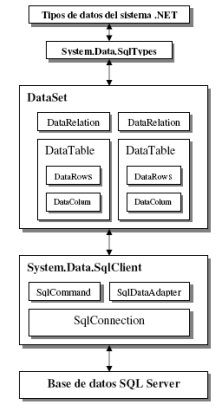


**Espacios de nombres y clases básicas de ADO.NET**

Hay seis espacios básicos de ADO.NET, pero solo veremos los concernientes a SQL Server y se recogen en la siguiente tabla. Además de estos espacios de nombres, cada nuevo proveedor de datos tiene su propio espacio de nombre. Como ejemplo, el proveedor de datos Oracle .NET añade un espacio de nombre de Microsoft.Data.OracleClient.



ADO.NET tiene tres tipos distintos de clases conocidas comúnmente como Disconnected, Shared y Data Providers. Las clases Disconnected proporcionan la estructura básica de ADO.NET. Un buen ejemplo de este tipo de clases es la clase DataTable. Los objetos de esta clase pueden guardar datos sin ninguna dependencia de un proveedor de datos específicos. Las clases Shared son las clases base para los proveedores de datos y se comparten entre todos los proveedores de datos. Las clases Data Provider están diseñadas para trabajar con distintos tipos de orígenes de datos. Se utilizan para ejecutar todas las operaciones de administración de datos sobre bases de datos específicas. Por ejemplo, el proveedor de datos sqlClient funciona sólo con la base de datos SQL Server. La figura. muestra un esquema de cómo encajan estos espacios de nombres y algunas de sus clases.

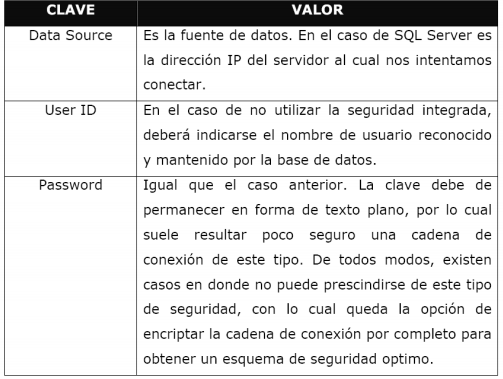


**Conexión**

A simple vista, las conexiones pueden parecer uno de los objetos más sencillos de utilizar en ADO.NET. Sin embargo, tomar las decisiones correctas al momento de su uso puede ser la diferencia entre una aplicación de altas prestaciones y otra de bajas. Existe una serie de premisas fundamentales que servirán para favorecer un rendimiento óptimo, las cuales se verán a lo largo de la guía de laboratorio. En primer lugar, el objeto Connection es un .NET Data Provider, y nos permite establecer la comunicación física entre nuestra aplicación y la base de datos. El .NET Framework posee la interfaz IDBConnection, la cual es implementada por las clases sqlConnection, oledbConnection y OracleConnection en sus versiones 1.x (1.0 y 1.1), y en el caso de su versión 2.0, es implementada por la clase DBConnection, que sirve como la base de los .NET Data Providers.

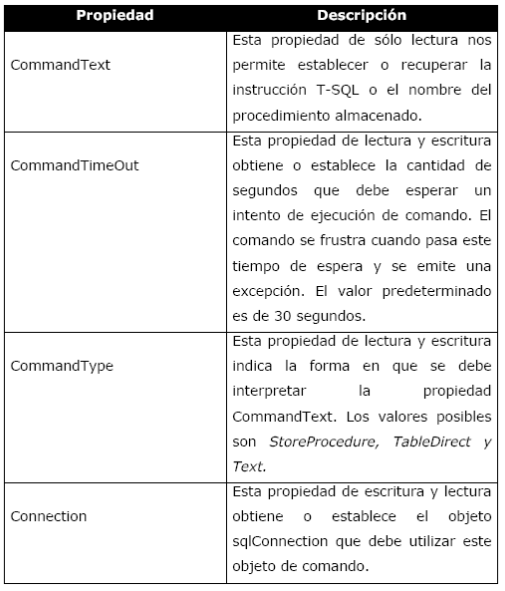
**Cadena de conexión**

Las conexiones establecen todos sus parámetros a través de la propiedad ConnectionString, la cual debe ser establecida de un modo correcto previo a su apertura. Esta propiedad es, en realidad, un conjunto de propiedades concatenados en una sola cadena de texto, en el formato “Propiedad=Valor;”. Muchas de estas propiedades que conforman la cadena son obligatorias, y otras poseen valores por defecto si no se establecen. Existen algunas propiedades bastantes específicas de cada proveedor de datos, y otras que pueden resultar sinónimos. A continuación veremos en la siguiente tabla las propiedades más comúnmente utilizadas en la cadena de conexión.

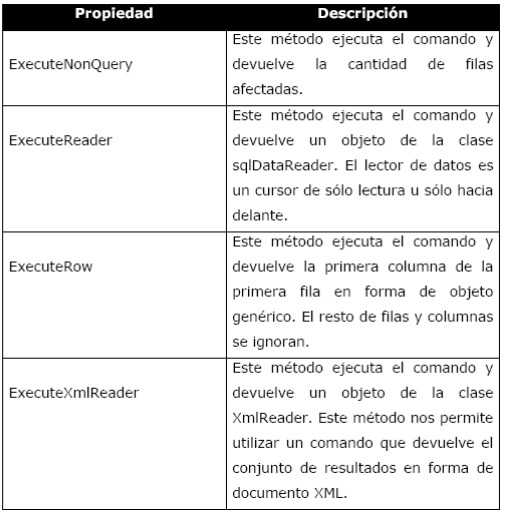


**Comandos**

Ya hemos visto que las conexiones nos brindan el vínculo entre las bases de datos y las aplicaciones, pero para poder establecer y ejecutar instrucciones necesitamos de los comandos, los cuales llamaremos indistintamente como xxxCommand. Una de las propiedades fundamentales de los comandos es CommandText que, como veremos más adelante, puede tener sentencias SQL o nombres de objetos de las bases de datos. Otra propiedad fundamental es Connection, a la cual le asignaremos una instancia de tipo xxxConnection. Las clases xxxCommand tienen constructores sobrecargados que nos permiten declarar, instancias e inicializar nuestros objetos en una sola línea de forma alternativa. Primero vamos a examinar algunas propiedades útiles de la clase sqlCommand que incluye la siguiente tabla:



Ahora vamos a examinar los diversos métodos Execute que se pueden llamar en un objeto Command.



**SQL Native Client OLE DB Provider**

El proveedor OLE DB de Microsoft SQL Native Client proporciona una interfaz OLE DB a las bases de datos de Microsoft SQL Server 2008. Este proveedor permite que las consultas distribuidas de SQL Server puedan consultar datos de instancias remotas de SQL Server.

Sintaxis: Provider=SQLOLEDB.1;Server=myServerName\theInstanceName; Database=myDataBase; Uid=myUsername;Pwd=myPassword;

**Dataset**

Un DataSet representa un conjunto completo de datos, incluyendo las tablas que contienen, ordenan y restringen los datos, así como las relaciones entre las tablas. Llenar el DataSet con tablas de datos de un origen de datos relacional existente mediante DataAdapter DataAdapter (Clase) Representa un conjunto de comandos SQL y una conexión de base de datos que se utilizan para rellenar el objeto DataSet y actualizar el origen de datos.

**DataAdapter (Clase)**

Representa un conjunto de comandos SQL y una conexión de base de datos que se utilizan para rellenar el objeto DataSet y actualizar el origen de datos.

**OleDbCommand (Clase)**

Representa una instrucción SQL o un procedimiento almacenado que se va a ejecutar en un origen de datos.

**DataGridView**

El control DataGridView proporciona una forma eficaz y flexible de mostrar datos en formato de tabla. Puede utilizar el control DataGridView para mostrar vistas de sólo lectura de una cantidad pequeña de datos o puede ajustar su tamaño para mostrar vistas modificables de conjuntos muy grandes de datos. DataGridView.DataSource (Propiedad) Obtiene o establece el origen de datos cuyos datos se están mostrando en el control DataGridView. Sintaxis: DataGridView1.DataMember = "MiTabla"