

Acceso a Datos en C#

Ing. Alejandro Calderón

Introducción a ADO.NET

ADO .NET es un conjunto de interfaces, clases, estructuras y enumeraciones que permiten el acceso a datos desde la plataforma .NET de Microsoft. Es una evolución lógica de la API tradicional de Microsoft conocida como ADO (ActiveX Data Object).

ADO .NET permite un modo de acceso desconectado a los datos, los cuales pueden provenir de múltiples fuentes de datos, con diferentes arquitecturas de almacenamiento.

ADO .NET soporta un modelo completo de programación y adaptación basado en el estándar XML.

Namespaces de ADO .NET

- ***System.Data***: Clases genéricas de ADO .NET. Integra la gran mayoría de clases que habilitan el acceso a los datos de la arquitectura .NET.
- ***System.Data.SqlClient***: Clases del proveedor de datos de SQL Server en su versión 7.0 y superior.
- ***System.Data.OleDb***: Clases del proveedor de datos de OleDb. Permite el acceso a proveedores .NET que trabajan directamente contra controladores basados en ActiveX de Microsoft.
- ***System.Data.SqlTypes***: Definición de los tipos de datos de SQL Server. Proporciona la encapsulación en clases de todos los tipos de datos nativos de SQL Server y sus funciones de manejo de errores, ajuste y conversión de tipos.
- ***System.Data.Common***: Clases base, reutilizables de ADO .NET. Proporciona la colección de clases necesarias para acceder a una fuente de datos.
- ***System.Data.Internal***: Integra el conjunto de clases internas de las que se componen los proveedores de datos.

Clases de *System.Data*

Dentro del espacio de nombres *System.Data* encontramos las siguientes clases compartidas que constituyen el eje central de ADO .NET:

- ***DataSet***: Almacén de datos por excelencia de ADO .NET. Representa una base de datos desconectada del proveedor de datos. Almacena tablas y sus relaciones.
- ***DataTable***: Un contenedor de datos. Estructurado como un conjunto de filas (*DataRow*) y columnas (*DataColumn*).
- ***DataRow***: Registro que almacena n valores. Representación en ADO. NET de una fila/tupla de una tabla de la base de datos.
- ***DataColumn***: Contiene la definición de una columna. Metadatos y datos asociados a su dominio.

- ***DataRelation***: Enlace entre dos o más columnas iguales de dos o más tablas.
- ***Constraint***: Reglas de validación de las columnas de una tabla.
- ***DataColumnMapping***: Vínculo lógico existente entre una columna de un objeto del *DataSet* y la columna física de la tabla de la base de datos.
- ***DataTableMapping***: Vínculo lógico existente entre una tabla del *DataSet* y la tabla física de la base de datos.

Además de estas clases, existe otro grupo de clases específicas de un proveedor de datos. Tienen una sintaxis con el formato XXXClase, donde “XXX” es un prefijo que determina el tipo de plataforma de conexión a datos, como ser *System.Data.SqlClient* y *System.Data.OleDb*.

Clase	Descripción
SqlCommand OleDbCommand	Clases que representan un comando SQL contra un sistema gestor de datos.
SqlConnection OleDbConnection	Clase que representa la etapa de conexión a un proveedor de datos. Encapsula la seguridad, pooling de conexiones, etc.
SqlCommandBuilder OleDbCommandBuilder	Generador de comandos SQL de inserción, modificación y borrado desde una consulta SQL de selección de datos.
SqlDataReader OleDbDataReader	Un lector de datos de sólo avance, conectado a la base de datos.

SqlDataAdapter OleDbDataAdapter	Clase adaptadora entre un objeto DataSet y sus operaciones físicas en la base de datos (select, insert, update y delete).
SqlParameter OleDbParameter	Define los parámetros empleados en la llamada a procedimientos almacenados.
SqlTransaction OleDbTransaction	Gestión de las transacciones a realizar en la base de datos.

Las Clases *Connection*

El primer paso obligado en un acceso a datos consiste en establecer una conexión con un almacén de datos. Esto se consigue a través de las clases *Connection* de ADO .NET, las cuales permiten la conexión a un origen de datos (ya sea una base de datos o no). El objeto *Connection* tiene dos versiones:

- *System.Data.SqlClient.SqlConnection*: Para un proveedor de datos de SQL Server.
- *System.Data.OleDb.OleDbConnection*: Para un proveedor de datos OLEDB.

En un objeto de tipo *Connection* se utilizan los métodos *Open()* y *Close()* para abrir y cerrar conexiones respectivamente, con el almacén de datos adecuado. Cuando ejecutamos el método *Open()* sobre un objeto *Connection*, se abrirá la conexión que se ha indicado en su propiedad *ConnectionString*, es decir, esta propiedad indicará la cadena de conexión que se va a utilizar para establecer la conexión con el almacén de datos correspondiente.

El constructor de las clases *Connection* se encuentra sobrecargado, en una de sus versiones recibe como parámetro una cadena de texto, la cual será la cadena de conexión que se asignará a la propiedad *ConnectionString*.

Ejemplo

```
string cadenaConexion =  
"Provider=Microsoft.ACE.OLEDB.12.0;Data Source='ejemplo.accdb'";  
OleDbConnection conexion = new OleDbConnection(cadenaConexion);  
conexion.Open();
```

Las Clases *Command*

Cuando ya hemos establecido una conexión con un almacén de datos, la siguiente operación lógica consiste en enviarle sentencias para realizar los distintos tipos de operaciones que normalmente hacemos con los datos. Para ello utilizaremos las clases *SqlCommand* y *OleDbCommand*.

El objeto *Command* nos permite ejecutar sentencias SQL sobre la fuente de datos a la que estamos conectados.

Un objeto *Command* debe ser creado a partir de una conexión ya existente, y debe tener una sentencia SQL definida.

Propiedades de *Command*

- ***CommandText***: Contiene una cadena de texto que indica la sentencia SQL que se va ejecutar sobre el origen de datos.
- ***CommandTimeout***: Tiempo de espera en segundos que se va a aplicar a la ejecución de un objeto Command. Su valor por defecto es 30 segundos.
- ***CommandType***: Indica el tipo de comando que se va a ejecutar en el almacén de datos (StoredProcedure, TableDirect o Text).
- ***Connection***: Devuelve el objeto SqlConnection u OleDbConnection utilizado para ejecutar el objeto Command correspondiente.
- ***Parameters***: Colección de parámetros que se pueden utilizar para ejecutar el objeto Command.

Métodos de *Command*

- ***CreateParameter***: Crea un parámetro para el que después podemos definir una serie de características específicas (tipo de dato, valor, tamaño, etc.) Devuelve un objeto de tipo *SqlParameter* u *OleDbParameter*.
- ***ExecuteNonQuery***: Ejecuta la sentencia SQL definida en la propiedad *CommandText* en la conexión definida en la propiedad *Connection*. La sentencia a ejecutar debe ser de un tipo que no devuelva un conjunto de registros, por ejemplo, *Update*, *Delete* o *Insert*. Devuelve la cantidad de filas afectadas.

- ***ExecuteReader***: Ejecuta la sentencia SQL definida en la propiedad *CommandText* en la conexión definida en la propiedad *Connection*. En este caso, la sentencia sí devuelve un conjunto de registros. El resultado es un objeto de tipo *SqlDataReader* u *OleDbDataReader*.
- ***ExecuteScalar***: Se utiliza cuando deseamos obtener la primera columna de la primera fila del conjunto de registros, el resto de datos no se toman en cuenta. Devuelve un objeto de tipo *Object*.
- ***Prepare***: Este método construye una versión compilada del objeto *Command* dentro del almacén de datos.

Ejemplo

```
string cadenaConexion =  
"Provider=Microsoft.ACE.OLEDB.12.0;Data Source='ejemplo.accdb'";  
OleDbConnection conexion = new OleDbConnection(cadenaConexion);  
conexion.Open();  
  
OleDbCommand comando = new OleDbCommand("SELECT * FROM persona", conexion);  
OleDbDataReader resultado = comando.ExecuteReader();  
conexion.Close();
```

Las Clases *DataReader*

Un objeto *DataReader* permite la navegación hacia delante de los registros devueltos por una consulta. Las clases que implementan el *DataReader* son *SqlDataReader* y *OleDbDataReader*.

Para obtener un objeto *DataReader* debemos ejecutar el método *ExecuteReader()* de un objeto *Command* basado en una consulta SQL.

Propiedades de *DataReader*

Las principales propiedades de un objeto *DataReader* son las siguientes:

- ***FieldCount***: Devuelve la cantidad de columnas (campos) presentes en la fila (registro) actual.
- ***IsClosed***: Devuelve True o False indicando si un DataReader está cerrado o no.
- ***Item***: Devuelve en formato nativo, el valor de la columna cuyo nombre le indicamos como índice en forma de cadena de texto.

Métodos de *DataReader*

- ***Close()***: Cierra el objeto *DataReader*.
- ***GetXXX()***: Son métodos que nos permiten obtener los valores de las columnas contenidas en el *DataReader*. Las XXX representan el tipo de dato. Por ejemplo: *GetBoolean()*, *GetInt32()*, *GetString()*, *GetChar()*, etc. Como parámetro debemos enviarle el número de columna que queremos recuperar (comenzando desde cero).
- ***NextResult()***: Desplaza el puntero actual al siguiente conjunto de registros (en caso de que se devuelva más de un conjunto de registros)
- ***Read()***: Desplaza el cursor actual al siguiente registro permitiendo obtener los valores del mismo por medio del *DataReader*. Devuelve True si existen más registros dentro del *DatarReader* y False si hemos llegado al final del conjunto de registros. Antes de comenzar a usar un *DataReader* debemos llamar al método *Read* para posicionarnos en el primer registro.

Ejemplo

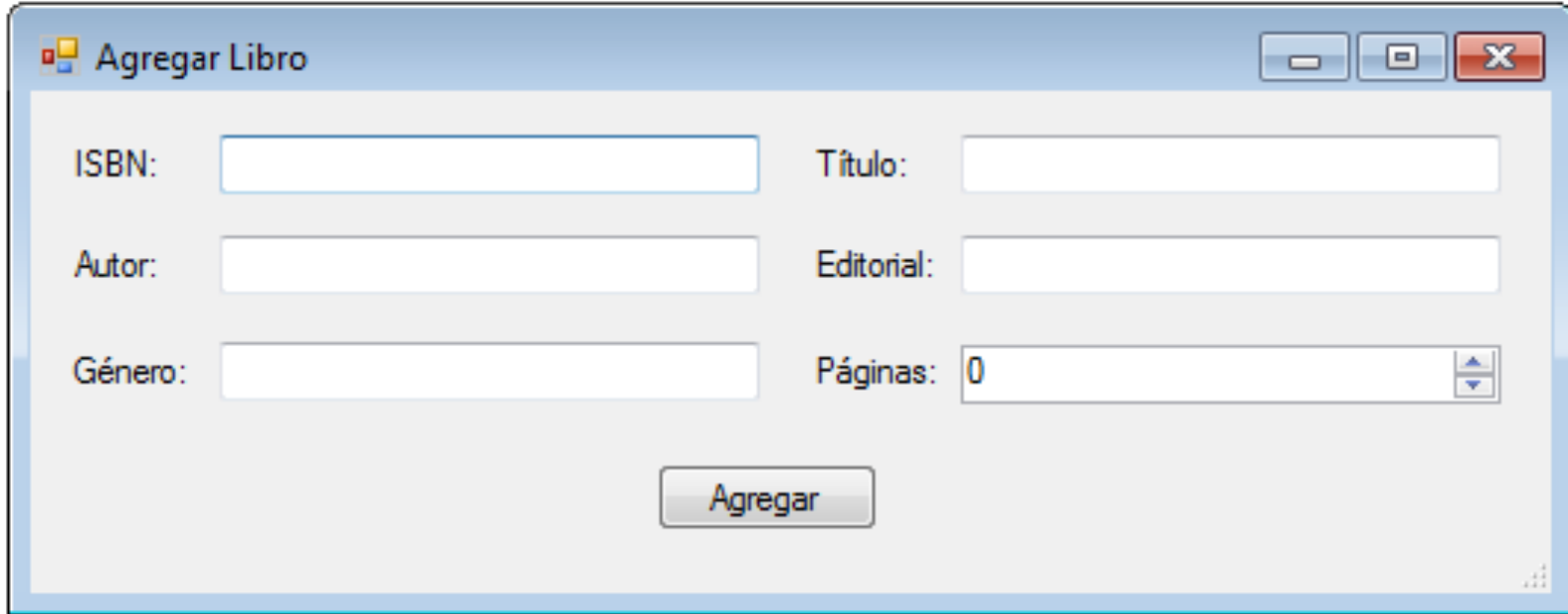
```
string cadenaConexion =  
"Provider=Microsoft.ACE.OLEDB.12.0;Data Source='ejemplo.accdb'";  
OleDbConnection conexion = new OleDbConnection(cadenaConexion);  
conexion.Open();  
  
string cadenaSQL = "INSERT INTO persona VALUES ('0801199501234', 'Ana', 'Ruiz')";  
OleDbCommand comando = new OleDbCommand(cadenaSQL, conexion);  
comando.ExecuteNonQuery();  
  
comando.CommandText = "SELECT * FROM persona";  
OleDbDataReader resultado = comando.ExecuteReader();  
  
while (resultado.Read())  
{  
    MessageBox.Show(resultado.GetString(0));  
    MessageBox.Show(resultado.GetString(1));  
    MessageBox.Show(resultado.GetString(2));  
}  
  
conexion.Close();
```

Ejercicio

The image shows a graphical user interface for a library application. The window is titled 'Libros' and has standard Windows-style window controls (minimize, maximize, close). On the left side, there are three buttons: 'Agregar libro', 'Modificar libro', and 'Borrar libro'. In the center, there is a list box containing the ISBN '0307951189', which is currently selected. To the right of the list box, the details for the selected book are displayed: ISBN: 0307951189, Título: Juego de Tronos, Autor: George R. R. Martin, Editorial: Vintage, Género: Fantasía, and Páginas: 800.

ISBN	Details
0307951189	ISBN: 0307951189 Título: Juego de Tronos Autor: George R. R. Martin Editorial: Vintage Género: Fantasía Páginas: 800

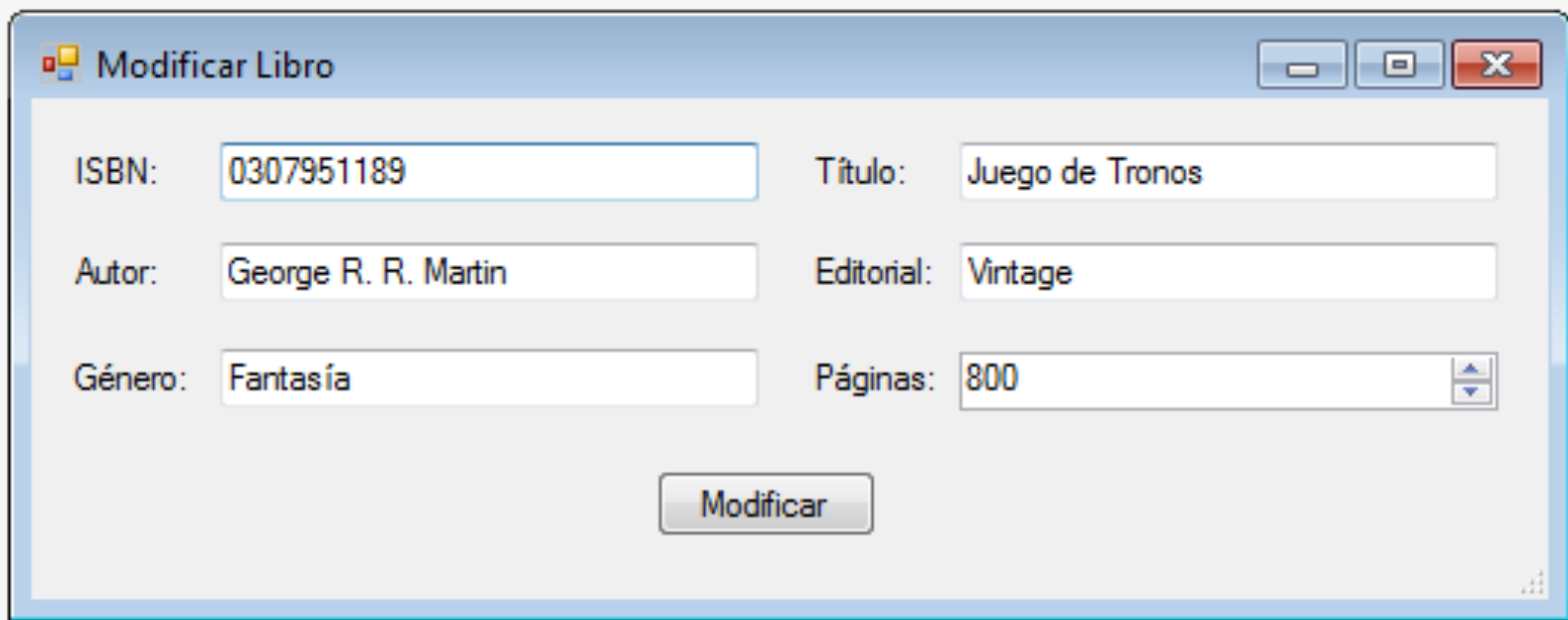
Ejercicio



A screenshot of a Windows-style dialog box titled "Agregar Libro". The dialog box has a light blue title bar with standard minimize, maximize, and close buttons. The main area is light gray and contains six input fields arranged in two columns. The left column has fields for "ISBN:", "Autor:", and "Género:". The right column has fields for "Título:", "Editorial:", and "Páginas:". The "Páginas:" field is a spinner box with the value "0". Below the input fields is a single button labeled "Agregar".

ISBN:	<input type="text"/>	Título:	<input type="text"/>
Autor:	<input type="text"/>	Editorial:	<input type="text"/>
Género:	<input type="text"/>	Páginas:	<input type="text" value="0"/>

Ejercicio



Modificar Libro

ISBN:	0307951189	Título:	Juego de Tronos
Autor:	George R. R. Martin	Editorial:	Vintage
Género:	Fantasía	Páginas:	800

Modificar

La Clase *DataSet*

La clase *DataSet* es una clase común de ADO .NET, lo que significa que se utiliza para cualquier tipo de proveedor de datos y no existen versiones particulares *SqlClient* u *OleDb*.

Un objeto de tipo *DataSet* representa la arquitectura de la base de datos completa, basada en un esquema XML, por lo que no depende de un fabricante específico. Un *DataSet* almacena los datos en una memoria caché desconectada. Se dice que un *DataSet* es almacenamiento pasivo, es decir, no se ve afectado por los cambios subyacentes en la base de datos.

Cada tabla de la base de datos está representada dentro del *DataSet* en la propiedad *Tables* que es de tipo *DataTable*. Un *DataTable* está compuesto por *DataRow*s (filas) y estos a su vez están compuestos por *DataColumns* (columnas).

Para poder inicializar las tablas de un *DataSet* debemos hacer uso de un objeto de tipo *DataAdapter*, el cual puede ser de dos tipos: *SqlDataAdapter* u *OleDbDataAdapter*. El objeto *DataAdapter* se construye enviándole como parámetro una cadena que representa la consulta que se va a ejecutar, y que va a rellenar de datos el *DataSet*.

Del objeto *DataAdapter* utilizaremos el método *Fill()*, el cual recibe como parámetros el *DataSet* que se quiere rellenar de información y un nombre para la tabla que se creará dentro del *DataSet*.

Propiedades de *DataSet*

- ***CaseSensitive***: Indica si las comparaciones de texto dentro de las tablas distinguen entre mayúsculas y minúsculas. Por defecto tiene valor falso.
- ***DataSetName***: Establece o devuelve el nombre del *DataSet*.
- ***HasErrors***: Devuelve un valor lógico para indicar si existen errores dentro de las tablas de un *DataSet*.
- ***Relations***: Devuelve una colección de objetos de tipo *DataRelation* que representan las relaciones existentes entre las tablas del *DataSet*.
- ***Tables***: Devuelve una colección de objetos de tipo *DataTable* que representan las tablas del *DataSet*.

Métodos de *DataSet*

- ***Clear()***: Elimina todos los datos almacenados en el objeto *DataSet*, vaciando todas las tablas contenidas en el mismo.
- ***AcceptChanges()***: Confirma todos los cambios realizados en las tablas y relaciones contenidas en el objeto *DataSet*, o bien los últimos cambios que se han producido desde la última llamada a *AcceptChanges*.
- ***GetChanges()***: Devuelve un objeto *DataSet* que contiene todos los cambios realizados desde que se cargó el *DataSet* o desde la última llamada a *AcceptChanges*.
- ***HasChanges()***: Devuelve verdadero o falso para indicar si se han realizado cambios al contenido del *DataSet* desde que fue cargado o desde la última llamada a *AcceptChanges*.
- ***RejectChanges()***: Abandona todos los cambios realizados en las tablas del *DataSet* desde que fue cargado o desde la última llamada a *AcceptChanges*.
- ***Merge()***: Toma el contenido de un *DataSet* y los mezcla con los de otro *DataSet*.

Las Clases *DataAdapter*

Las clases *DataAdapter* (*SqlDataAdapter* y *OleDbDataAdapter*) sirven como puente entre el origen de datos y el *DataSet*, ya que permiten cargar el *DataSet* con información de la fuente de datos, y posteriormente, actualizar el origen de datos con la información del *DataSet*.

La clase *DataAdapter* tiene cuatro propiedades de tipo *Command*:

- ***InsertCommand***: Se utiliza para inserción de datos.
- ***SelectCommand***: Se utiliza para ejecutar sentencias *select*.
- ***UpdateCommand***: Se utiliza para realizar modificación de datos.
- ***DeleteCommand***: Se utiliza para realizar eliminación de datos.

El Método *Fill*

Un método destacable de las clases *DataAdapter* (*SqlDataAdapter* y *OleDbDataAdapter*) es el método *Fill()*, el cual ejecuta el comando de selección asociado a la propiedad *SelectCommand*. El resultado de esta consulta se carga en el objeto *DataSet* que le pasamos como parámetro.

Data Binding: Enlace de Datos a Controles

Data Binding es el mecanismo proporcionado por la plataforma .NET para enlazar objetos contenedores de datos con los controles de un formulario, para poder realizar operaciones automáticas de navegación y edición.

Existen dos tipos de *Data Binding*:

- Enlace simple (*Simple Data Binding*): Este tipo de enlace consiste en una asociación entre un control que puede mostrar un único dato y el objeto contenedor de datos (ejemplo: *TextBox*).
- Enlace complejo (*Complex Data Binding*): En este enlace, el control que actúa como interfaz o visualizador de datos, dispone de la capacidad de mostrar todos los datos del objeto contenedor (ejemplo: *DataGrid*).

Elementos del proceso *Data Binding*

- ***Binding***: Clase que permite crear un enlace (*binding*) para un control, indicando la propiedad del control que mostrará los datos, el *DataSet* del que se extraerá la información, y el nombre de la tabla – columna, cuyos datos pasarán a la propiedad del control.
- ***DataBindings***: Colección de que disponen los controles, con la información de enlaces a datos. Gracias a su método *Add()* podemos añadir un objeto *Binding* para que el control muestre los datos que indica el enlace.
- ***BindingContext***: Propiedad de la clase *Form*, que representa el contexto de enlace a datos establecido en los controles del formulario, es decir, toda la información de enlaces establecida entre los controles y objetos proveedores de datos. Devuelve un objeto de tipo *BindingManagerBase*.
- ***BindingManagerBase***: Objeto que se encarga de administrar un conjunto de objetos de enlace, por ejemplo, los de un formulario, obtenidos a través del *BindingContext* del formulario.