Ado.net

ADO.NET es un conjunto de clases que exponen servicios de acceso a datos para programadores de .NET Framework. ADO.NET ofrece abundancia de componentes para la creación de aplicaciones de uso compartido de datos distribuidas. Es una librería de clases que constituye una parte integral de .NET Framework y proporciona acceso a datos relacionales, XML y de aplicaciones. También hoy en día puede conectarse a bases de datos **NOSql** (**N**ot **O**nly **Sql**), que son bases de datos de estructura no relacional.

ADO.NET satisface diversas necesidades de desarrollo, como la creación de clientes front-end de base de datos y objetos empresariales de nivel medio (lógica de negocio), que utilizan aplicaciones, herramientas, lenguajes o exploradores de Internet.

https://docs.microsoft.com/es-es/dotnet/framework/data/adonet/

Componentes de ADO.NET

Los dos componentes principales de ADO.NET para el acceso a los datos y su manipulación son los proveedores de datos .NET Framework y DataSet.

Nota Importante: si bien mencionaremos básicos de DataSet, DataAdapter y DataTable, sólo se recomienda su uso en caso en forma esporádica y para tareas de administración de la base de datos que uno deba hacer desde código C#, por ejemplo, necesitar recuperar parte del esquema de la base de datos (parte de la estructura de la base de datos), o necesitar crear con permisos adecuados una tabla nueva en una base de datos. Nos centraremos en esta clase y en la siguiente en los objetos Connection , Command , Parameter y DataReader , adecuados para una aplicación en capas que consume datos.

En el desarrollo con arquitectura en capas o con APIs de acceso a datos (microservicios), es recomendable crear la lista de entidades correspondientes a la lógica de negocio de la aplicación, que recuperen los datos desde un DataReader de ADO.Net en la capa de acceso a datos y los pasa a la capa de negocio en una lista de objetos, de esta forma la capa de negocio queda totalmente libre de los objetos ADO.Net, y no estaría perdiendo performance o haciendo lenta la aplicación o API, dado que el trabajo con DataSet/DataTable es pesado y genera sobrecargas.

Proveedores de datos .NET Framework

Los proveedores de datos .NET Framework son componentes diseñados explícitamente para la manipulación de datos y el acceso rápido a datos de solo lectura y solo avance.

El objeto Connection proporciona conectividad a un origen de datos.

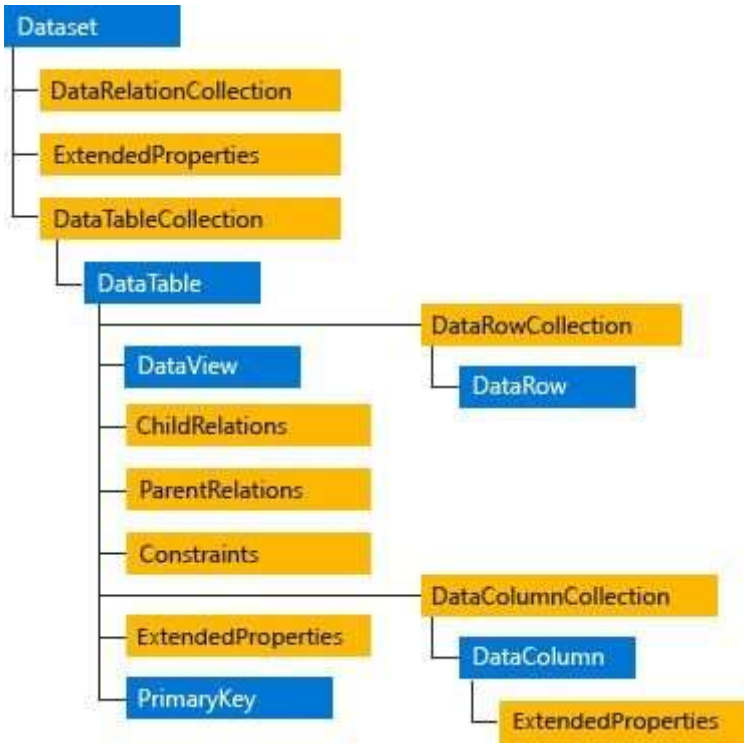
El objeto Command permite tener acceso a comandos de base de datos para devolver datos, modificar datos, ejecutar procedimientos almacenados y enviar o recuperar información sobre parámetros.

El objeto DataReader proporciona un flujo 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. DataAdapter utiliza objetos Command para ejecutar comandos SQL en el origen de datos tanto para cargar DataSet con datos y reconciliar en el origen de datos los cambios aplicados a los datos incluidos en el DataSet.

DataSet

El DataSet de ADO.NET está expresamente diseñado para el acceso a datos independientemente del origen de datos. Como resultado, se puede utilizar con múltiples y distintos orígenes de datos, con datos XML/JSON o para administrar datos locales de la aplicación.

DataSet contiene una colección de uno o más objetos DataTable formados por filas y columnas de datos, así como información sobre claves principales, claves externas, restricciones y de relación relacionada con los datos incluidos en los objetos DataTable. En el diagrama siguiente se ilustra la relación entre un proveedor de datos .NET Framework y un DataSet.



Arquitectura de ADO.NET

Nota: La serialización de un DataSet a formato JSON es un tema avanzado pero alternativo y conveniente para manejo de datos. Recomendamos este link: Serialize a DataSet

Elegir un DataReader o un DataSet

A la hora de decidir si su aplicación debe utilizar un DataReader o un DataSet, debe tener en cuenta el tipo de funcionalidad que su aplicación requiere. Use un DataSet para hacer lo siguiente:

- Almacene datos en la memoria caché de la aplicación para poder manipularlos. Si solamente necesita leer los resultados de una consulta, el DataReader es la mejor elección, es más ágil y rápido.
- Utilizar datos de forma remota entre un nivel y otro o desde un servicio Web XML/JSON.
- Interactuar con datos dinámicamente, por ejemplo para enlazar con un control de Windows Forms o para combinar y relacionar datos procedentes de varios orígenes de datos.
- Realizar procesamientos exhaustivos de datos sin necesidad de tener una conexión abierta con el origen de datos, lo que libera la conexión para que la utilicen otros clientes.

Si no necesita la funcionalidad proporcionada por el DataSet, **puede mejorar el rendimiento de su aplicación si utiliza el DataReader** para devolver sus datos de solo avance y de solo lectura. Aunque DataAdapter utiliza DataReader para rellenar el contenido de un DataSet , al utilizar el DataReader puede mejorar el rendimiento porque no usará la memoria que utilizará el DataSet, además de evitar el procesamiento necesario para crear y rellenar el contenido de DataSet.

Si llegara a necesitar más información sobre DataSet, se recomienda este link: Objetos DataSet, DataTable y DataView

LINQ to DataSet

Proporciona capacidades de consulta y comprobación de tipo en tiempo de compilación de los datos almacenados en caché de un objeto DataSet. Permite escribir consultas en uno de los lenguajes de desarrollo de .NET Framework.

LINQ to SQL

Admite consultas en un modelo de objetos asignado a las estructuras de datos de una base de datos relacional sin utilizar un modelo conceptual intermedio. Cada tabla se representa mediante una clase distinta, acoplando de manera precisa el modelo de objetos al esquema de la base de datos relacional. LINQ to SQL convierte las consultas de Language-Integrated Query del modelo de objetos a Transact-SQL y las envía a la base de datos para su ejecución. Cuando la base de datos devuelve los resultados, LINQ to SQL los vuelve a traducir a objetos.

ADO.NET Entity Framework

Está diseñado para permitir que los desarrolladores creen aplicaciones de acceso a los datos programando en un modelo de aplicación conceptual en lugar de programar directamente en un esquema de almacenamiento relacional. El objetivo es reducir la cantidad de código y mantenimiento que se necesita para las aplicaciones orientadas a datos.

Servicios de datos de WCF

Describe cómo se usa Servicios de datos de WCF para implementar servicios de datos en web o en una intranet. Los datos se estructuran como entidades y relaciones de acuerdo a

las especificaciones de Entity Data Model. Los datos implementados en este modelo se pueden direccionar mediante el protocolo HTTP estándar.

XML y ADO.NET

ADO.NET aprovecha la eficacia de XML para proporcionar acceso a datos sin conexión. ADO.NET fue diseñado teniendo en cuenta las clases de XML incluidas en .NET Framework; ambos son componentes de una única arquitectura.

ADO.NET y las clases de XML incluidas en .NET Framework convergen en el objeto DataSet. DataSet se puede rellenar con datos procedentes de un origen XML, ya sea éste un archivo o una secuencia XML. DataSet se puede escribir como XML conforme al consorcio World Wide Web (W3C), que incluye su esquema como esquema lenguaje de definición de esquemas XML, independientemente del origen de los datos incluidos en DataSet. Puesto que el formato nativo de serialización del DataSet es XML, es un medio excelente para mover datos de un nivel a otro, por lo que DataSet es idóneo para usar datos y contextos de esquemas de interacción remota desde y hacia un servicio Web XML.

Uso básico de Sql Managment Studio

SQL Server Management Studio (SSMS) es un entorno integrado para obtener acceso, configurar, administrar y desarrollar todos los componentes de SQL Server. SSMS combina un amplio grupo de herramientas gráficas con una serie de editores de script enriquecidos que permiten a desarrolladores y administradores de todos los niveles obtener acceso SQL Server.

SSMS combina las características del Administrador corporativo, el Analizador de consultas y Analysis Manager, herramientas incluidas en versiones anteriores de SQL Server, en un único entorno. Además, SSMS funciona con todos los componentes de SQL Server, como Reporting Services e Integration Services. De este modo, los desarrolladores pueden disfrutar de una experiencia familiar y los administradores de bases de datos disponen de una herramienta única y completa que combina herramientas gráficas fáciles de usar con funciones avanzadas de scripting.

Es un frontend que permite conectarse a bases de datos de la familia SQL Server. De manera de poder trabajar y probar más fácilmente sus consultas. De todas maneras Visual Studio posee un entorno similar para poder ejecutar dichas queries.

Uso de cadenas de conexión (connection Strings)

Una cadena de conexión contiene información de inicialización que se transfiere como un parámetro desde un proveedor de datos a un origen de datos. La sintaxis depende del proveedor de datos y la cadena de conexión se analiza mientras se intenta abrir una conexión. Las cadenas de conexión que usa Entity Framework contienen la información que se emplea para conectar con el proveedor de datos ADO.NET subyacente que Entity Framework admite. También contienen información sobre los archivos del modelo y de asignación necesarios.

El proveedor de EntityClient utiliza la cadena de conexión al obtener acceso a los metadatos del modelo y de asignación y al conectar con el origen de datos. Se puede obtener acceso a la cadena de conexión o establecerse a través de la propiedad ConnectionString de

EntityConnection. La clase EntityConnectionStringBuilder se puede utilizar para construir mediante programación los parámetros de la cadena de conexión o tener acceso a ellos.

Las herramientas de Entity Data Model generan una cadena de conexión que se almacena en el archivo de configuración de la aplicación. ObjectContext recupera esta información de conexión automáticamente al crear consultas de objetos. Se puede tener acceso al elemento EntityConnection que usa una instancia de ObjectContext desde la propiedad Connection.

Parámetros de la cadena de conexión

El formato de una cadena de conexión es una lista de pares de parámetros de clave y valor delimitados por punto y coma:

keyword1=value; keyword2=value;

El signo igual (=) asocia cada palabra clave a su valor. Las palabras clave no distinguen entre mayúsculas y minúsculas y los espacios entre los pares clave-valor se omiten. Sin embargo, los valores pueden distinguir entre mayúsculas y minúsculas, en función del origen de datos. Los valores que contengan un punto y coma, caracteres de comilla sencilla o caracteres de comilla doble deben colocarse entre comillas dobles.

Para detalles de cadenas de conexión de todos los orígenes de datos relacionales y no relacionales, consulte el siguiente link: https://www.connectionstrings.com/

Ejemplo de Connection String con SQL Server

```
Server=myServerAddress;Database=myDataBase;User Id=myUsername;

Password=myPassword;
```

Ejemplo de Connection String con Oracle

```
Data Source=MyOracleDB;User Id=myUsername;Password=myPassword;
Integrated Security=no;
```

En caso de No Usar Archivos TNS, se puede utilizar su mismo formato:

```
SERVER=(DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(HOST=MyHost)(PORT=MyPort))(CON
NECT_DATA=(SERVICE_NAME=MyOracleSID))); uid=myUsername;pwd=myPassword;
```

Ejemplo de Connection String con MySQL Estándar:

```
Server=myServerAddress;Database=myDataBase;Uid=myUsername;Pwd=myPassword;
```

Especificando Puerto TCP/IP:

```
Server=myServerAddress;Port=1234;Database=myDataBase;Uid=myUsername;
Pwd=myPassword;
```