Universidad Abierta Interamericana

Campus Virtual UAIOnline

LENGUAJES DE ÚLTIMA GENERACIÓN

ACCESO A DATOS

UNIDAD 2/CLASE ACTUAL:
ACCESO A DATOS /CLASE

Autor de contenidos:
Mauricio Prinzo

PRESENTACIÓN

Vamos hablar del metodo desconectado de ADO.Net

1. ENTORNO DESCONECTADO

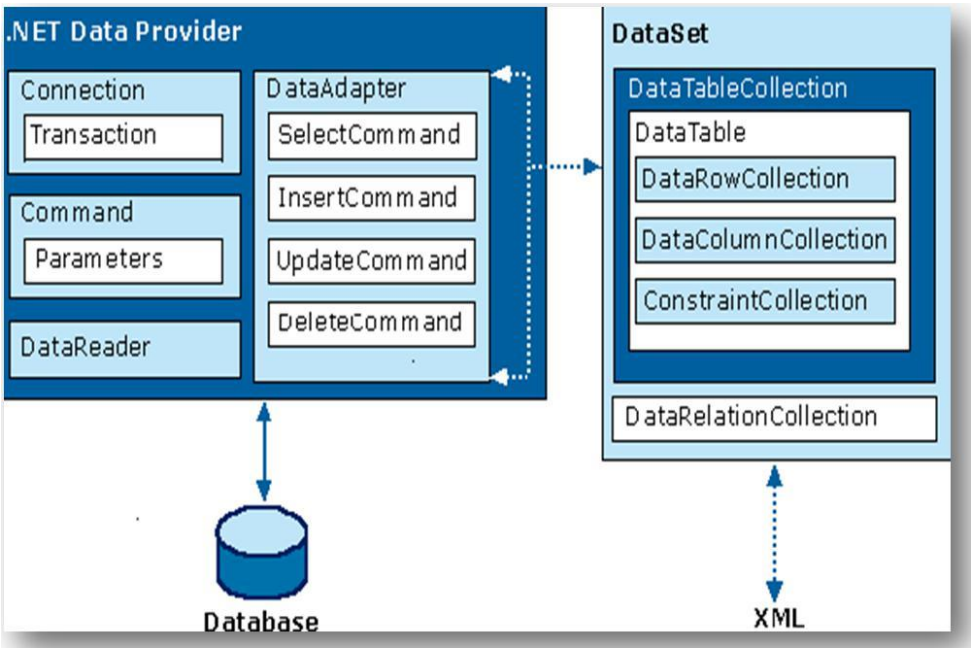
Un entorno desconectado es aquel en el que los datos pueden modificarse de forma independiente y los cambios se escriben posteriormente en la base de datos

Ventajas:

- Las conexiones se utilizan durante el menor tiempo posible, permitiendo que menos conexiones den servicio a más usuarios
- Un entorno desconectado mejora la escalabilidad y el rendimiento de las aplicaciones

Inconvenientes:

- Los datos no siempre están actualizados
- Pueden producirse conflictos de cambios que deben solucionarse



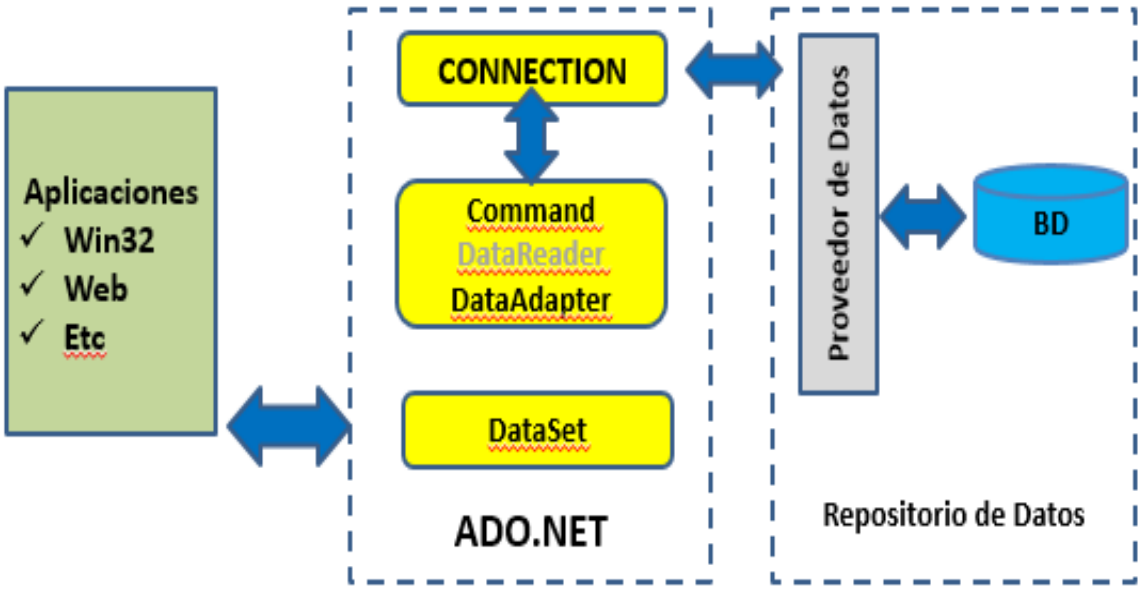
2. MODO DESCONECTADO

Nos encontramos en un entorno desconectado cuando el modelo de datos está conectado para obtener los datos y luego se puede desconectar sin perder los datos.

Uno de los principales problemas que tiene es la **sincronización** de los datos, puesto que si alguien cambió los datos mientras se estaba usando no sabremos cual es el valor definitivo.

Es decir, al trabajar desconectado dos usuarios se llevan una copia de los datos, hacen modificaciones por separado y luego deben actualizar la base de datos. La pregunta que seguramente se harán es qué pasa si un usuario cambia el precio de $10 a $12 y otro hace un cambio de $10 a $14, ¿cuál es el valor final? En esta asignatura no trabajaremos en la resolución de este tipo de problemas pero debe usted saber que este se presenta cuando trabajamos en un entorno desconectado.

Esquema modo Desconectado



2.1 Objeto Dataset

El objeto **DataSet** no forma parte del modelo ADO.net pero se complementan ya que puede funcionar como una base de datos temporal. Es una evolución del viejo *recordset* de la versión anterior incorporando la posibilidad de contener varias tablas y relaciones entre dichas tablas.

Contiene **datatables** que son tablas vinculadas al modelo de datos y **dataRelation** que son las relaciones entre las tablas.

A su vez, el **datable** está formado por **datarows** y estas a su vez por **datacolumns**.

La ventaja del **DatSet** es que está formado por colecciones enlazadas que se pueden recorrer fácilmente con la estructura **foreach**.

```
foreach (DataRow item in Ds.tables[0].Rows)
{ .. }
```

El **DataSet** que funciona desconectado puede crearse como un modelo desvinculado de la base de datos.

```
DataSet Ds = new DataSet();
DataTable Dt = new DataTable();
Dt.TableName=”Persona”;
//Creo la columna Id
   DataColumn ColId ;

//Completo la columna Id
      ColId = new DataColumn("ID", typeof(int));

//Creo la columna Nombre
       DataColumn ColNombre = new As DataColumn();
//Completo la columna ColNombre
ColNombre.ColumnName =”Nombre”;
ColNombre.DataType= typeof(string);
COlNombre.MaxLength = 50;

//Agrego las columnas a la tabla
       Dt.Columns.Add(ColId);
```

```
        Dt.Columns.Add(ColNombre);

//Creo una fila con los datos en las columnas
        DataRow Dr;
        Dr = Dt.NewRow();
        Dr["Id"] = 1:
        Dr["Nombre"] = "Juan";
        Dt.Rows.Add(Dr);

        Dr = Dt.NewRow();
        Dr["ID"] = 2:
        Dr["Nombre"] = "Pedro";
        Dt.Rows.Add(Dr);

//Determino la clave primaria

      Dt.PrimaryKey = new DataColumn[] { Id };
      Ds.Tables.Add(Dt);

        this.cmbDataset.DataSource = Ds.Tables[0];
        this.cmbDataset.DisplayMember = "Nombre" ;
        this.cmbDataset.ValueMember = "Id";
```

Notemos que las últimas tres líneas muestran como desde un **Dataset** se puede popular un combobox sin necesidad de recorrer toda la tabla. Algo similar se puede lograr si utilizamos una Datagrid.

Cuando trabajamos en objetos, pasaremos tanto a la capa de presentacion listas de objetos, osea mapearemos los datos a listas.

Resumiendo, los pasos para crear un dataset sin una base de datos son:

1- Crear un objeto **Dataset**
2- Crear un **Datatable**
3- Crear el o los objetos **datacolumn** que se necesiten determinando el nombre del campo y el tipo de datos.
4- Agregar al **Datatable**
5- Llenar la colección de claves de la tabla con los campos que la componen
6- Agregar el objeto **Datatable** al **DataSet**
7- De existir, crear todas las relaciones

2.2 OBJETO SQLADAPTER

Este objeto es el nexo entre el modelo ADO.net con el objeto **Dataset**. Y por supuesto hay un tipo **DataAdapter** para cada proveedor de datos. si bien es frecuente nuestra referencia al objeto sqlDataAdapter existe también un **oldbdataadapter**Seguramente, en otras asignaturas habrá estudiado acerca de los constructores de los objetos. ¿Recuerdar el método **new**?.

El **dataAdapter**, tiene una sobrecarga en su constructor, entre sus opciones puede no tener argumentos o agregar argumentos. Veremos cómo aprovechar los comandos en modo desconectado en los siguientes ejemplos.

Una rutina donde utilizamos el **SqlDataAdapter** para llenar un **dataset**

```
DataSet Ds = new DataSet();
SqlCommand oCmd = new SqlCommand():
SqlDataAdapter oDa = new SqlDataAdapter();
oCmd.Transaction = TX;
oCmd.Connection = oCnn;
oCmd.CommandType = CommandType.Text;
oCmd.CommandText = Query;
oDa.SelectCommand = oCmd;
oDa.Fill(Ds);
oDa.Dispose();
oCmd.Dispose();
Return Ds;
```

Otro ejemplo

```
DataSet ds = new DataSet();
 SqlCommand oCmd = new SqlCommand();
 oCmd.Transaction = TX;
 oCmd.Connection = oCnn;
 oCmd.CommandType = CommandType.Text;
 oCmd.CommandText = Query;
 ds = new DataSet();
 SqlDataAdapter da = new SqlDataAdapter(oCmd);
 //con argumentos
 da.Fill(ds);
 da.Dispose();
 oCmd.Dispose();
 return ds;
```

En el código, las líneas nuevas e importantes son:

```
SqlDataAdapter oDa = new SqlDataAdapter();
oDa.SelectCommand = oCmd;
```

Crea el **sqldataadapter** y lo relaciona con el comando. Esto mismo se podría escribir de la misma manera pero usando el constructor sobrecargado del objeto.

```
sqlDataAdapter Da = new sqlDataAdapter(Ocmd);
```

Le solicitamos que profundice sus conocimientos acerca de ADO.net, con la lectura de la bibliografia.