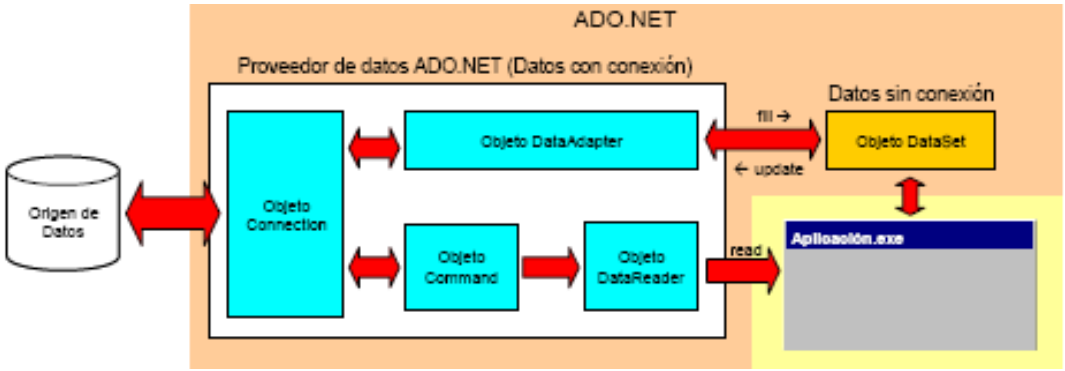
PRINCIPIO DE EJERCICIO MODO DESCONECTADO GUIADO

En el ejemplo tratado anteriormente se trabajaba en modo conectado. Ahora vamos a trabajar con el modo Desconectado.

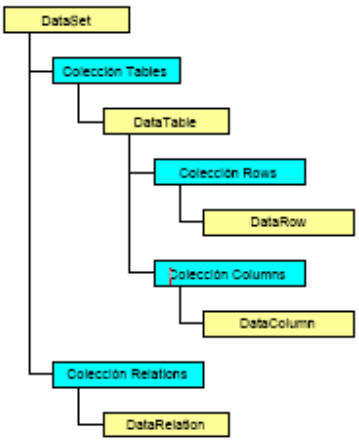
Esto es:

La conexión sólo es necesario establecerla cuando se descarga información del origen de datos.

Se tomarán los datos con un elemento **DataAdapter** y se rellenará un elemento denominado **DataSet**. Este objeto es un elemento que se compone de tablas (**DataTables**) que pueden estar relacionadas a través de elementos (**DataRelations**). Cada tabla (**DataTable**) del **DataSet** está formada por un conjunto de filas (**DataRows**) y un conjunto de campos o columnas (**DataColumns**).



Los datos con los que vamos a trabajar estarán guardados en la memoria caché del cliente en un objeto **DataSet**.



El modo desconectado se utiliza cuando:

- Se necesita modificar los datos frecuentemente.
- Es necesario que los datos estén mucho tiempo en memoria (por ejemplo en aplicaciones Windows Forms).
- Cuando no siempre es posible estar conectado al origen de datos (por ejemplo en aplicaciones móviles).

Vamos a hacer el mismo ejemplo del otro día pero en modo desconectado.

1º Tendremos que importar dos espacios de nombres: **System**.**Data SqlClient**, donde ya vimos en el ejemplo anterior que se encuentran todas las clases para acceder a origenes de datos de SQL Server. Y en este caso también **System**. **Data**. Porque los DataSet, Datatables,… se encuentran en este espacio de nombres.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Data;
using System.Data.SqlClient;
```

2º Crearemos un método para crear y abrir la **conexión**, crear el **adaptador** y el **dataset**.

Antes de nada, como muchos de los objetos van a ser utilizados en varios procedimientos tendrán que ser definidos en la Partial Class.

```
public partial class _Default : System.Web.UI.Page
{

    SqlConnection conn;
    SqlDataAdapter dalibros;
    DataSet dslibros;
    DataTable dtlibros;

protected void abrirconexion()
    {
       String cadenaconexion = "Data Source=.\\SQLEXPRESS;Initial
       Catalog='PUBS';Integrated Security=True";
        conn = new SqlConnection(cadena);
        conn.Open();
        String comando = "Select * from titles";
        SqlDataAdapter dalibros = new SqlDataAdapter(comando, conn);
```

```
        DataSet dslibros = new DataSet();
        dalibros.Fill(dslibros, "libros");
        dtlibros = dslibros.Tables[0];
        conn.Close();

    }
```

Hasta aquí hemos creado la conexión y la hemos abierto.

```
String cadenaconexion = "Data Source=.\\SQLEXPRESS;Initial
Catalog='PUBS';Integrated Security=True";
conn = new SqlConnection(cadena);
conn.Open();
```

Hemos creado el adaptador (es como el **SqlCommand**, pero para trabajar en modo desconectado).

```
String comando = "Select * from titles";
SqlDataAdapter dalibros = new SqlDataAdapter(comando, conn);
```

Creamos el **Dataset** y generamos la tabla dentro de él. Esto se hace a través del método **Fill** del Adaptador. Al cual se le pasan dos parámetros, el primero es el nombre del Dataste que vas a rellenar y el segundo el nombre con el que vas a reconocer dentro del dataset a la tabla que estas creando.

```
DataSet dslibros = new DataSet();
dalibros.Fill(dslibros, "libros");
```

Si nos fijamos el objeto **DataSet** no es SqldataSet. Esto es debido a que es un objeto “abstracto”: está desligado a cualquier origen de datos. Lo mismo ocurrirá con sus subelementos **DataTable**, **DataRow**, **DataColumn**,…

Por último para trabajar más cómodamente en la aplicación se va a definir un elemento **DataTable**, si no todas las operaciones que debamos hacer desde la tabla deberíamos hacerlas desde **dslibros.Tables[0**], de esta forma las haremos desde **dtlibros**.

```
        dtlibros = dslibros.Tables[0];
```

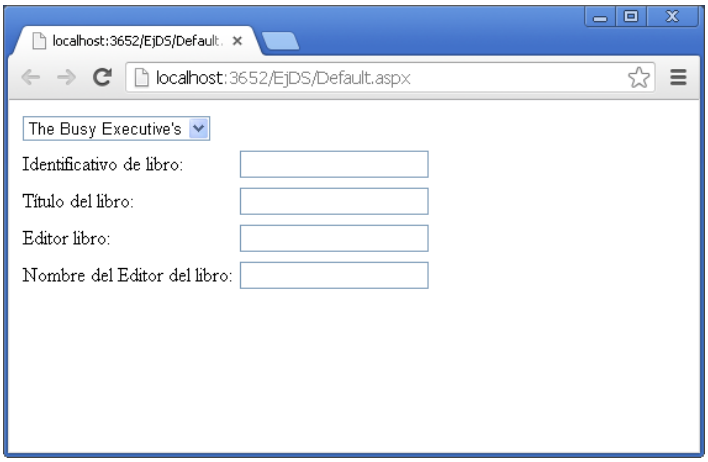
Lo que vamos a hacer ahora es en el evento **Page_Load**, es decir, nada mas cargar la aplicación, cargar en la lista desplegable los títulos de los libros.

Como ya tenemos los libros cargados en el objeto **dtlibros**, accederemos a sus filas a través de **dtlibros.Rows**. Recorreremos el conjunto de filas que tenemos en mi **DataTable dtlibros** y cargaremos el campo que nos interesa de las filas **dr[“nombrecampo”].**

```
protected void Page_Load(object sender, EventArgs e)
```

```
{
    abrirconexion();

    foreach (DataRow dr in dtlibros.Rows)
    {
          ddllibros.Items.Add(new
          ListItem(dr["title"].ToString(),dr["title_id"].ToString()));
    }
}
```



3º El siguiente paso es que al seleccionar el título aparezcan sus datos en los cuadros de texto. Esto lo podemos hacer de dos formas.

3.1.- Para buscar varias filas que cumplan un criterio, se puede utilizar el método **Select**, sobre la **dataTable**.

3.2.- Para buscar una sola fila que cumpla un criterio, es decir buscar sobre algún campo clave se utiliza el método **Find** sobre el conjunto de **filas** de una **datatable**.

Método Select : Este método devuelve un conjunto de filas, es decir un array de tipo DataRow. Al método select se le pasa un string con la condición que se está buscando, es decir, lo que pondríais detrás de la cláusula WHERE en una consulta.
Como segundo método se puede poner el orden buscado, lo que pondríais después de la cláusula ORDER BY en una consulta.

```
      DataRow[] filas;
    filas=dtlibros.Select("title='" + ddllibros.SelectedItem.Text + "'");
```

Luego se accede al conjunto de filas generado para acceder a cada uno de los campos y listo.

Como en nuestro caso solo va a encontrar un libro, sabemos que el libro se encontrará en la primera posición del array de filas, es decir, en la posición 0.

```
        txtidlibro.Text = filas[0]["title_id"].ToString();
        txtlibro.Text = filas[0]["title"].ToString();
        txtideditor.Text = filas[0]["pub_id"].ToString();
```

```
protected void ddllibros_SelectedIndexChanged(object sender, EventArgs e)
{
     DataRow[] filas;
    filas=dtlibros.Select("title='" + ddllibros.SelectedItem.Text + "'");
    txtidlibro.Text = filas[0]["title_id"].ToString();
    txtlibro.Text = filas[0]["title"].ToString();
    txtideditor.Text = filas[0]["pub_id"].ToString();
}
```

Método Find : Este método devuelve una sola fila. Pero el método **Find** solo busca en campos que son clave, por tanto hay que definirlos en la **Datatable**. Esto se hace de la siguiente forma:

```
       DataColumn[] clave = new DataColumn[1];
       clave[0] = dtlibros.Columns["title_id"];
       dtlibros.PrimaryKey = clave;
```

Hemos creado uun objeto de tipo columna y le hemos dicho que este objeto va a ser el campo **title_id** de la **dataTable dtibros**. A continuación tenemos que decirle que esta columna va a ser la clave principal del objeto **DataTable dtlibros**.

En nuestro caso como la clave por la que buscamos es un solo campo, la busqueda se puede hacer directamente:

```
DataRow filaencontrada = dtlibros.Rows.Find(ddllibros .SelectedItem .Value);
```

Y presentar los datos:

```
txtidlibro.Text = filaencontrada["title_id"].ToString();
txtlibro.Text = filaencontrada["title"].ToString();
txtideditor.Text = filaencontrada["pub_id"].ToString();
```

Si la clave por la que buscar estuviese formada por más de un campo, habría que haber creado un array de valores, asignarle a este array los campos y buscar este array en el método find.

```
       Object[] valor=new Object[2];
       valor[0]=txtejemplo1.text;
       valor[1]=txtejemplo2.text;
       DataRow filaencontrada = dtlibros.Rows.Find(valor);
```

Para insertar, borrar y modificar registros se hará de la siguiente forma:

Insertar un registro en un DataTable (Método NewRow)

Para insertar un registro en una tabla se utilizará el método NewRow de un datatable. Después añadiremos los valores introducidos en los controles y por último se añadirá la nueva fila al conjunto de filas de la tabla.

```
DataRow dr = dtlibros.NewRow()
    dr["nombrecampo"] = txtcontrol1.Text;
    dr["nombrecampo "] = txtcontrol2.Text;
    dr["nombrecampo "] = txtcontrol3.Text;
    dtlibros.Rows.Add(dr);
```

Borrar un registro en un DataTable (Método Delete)

Deberemos localizar el registro a borrar y una vez localizado se le aplicará el método delete sobre ese registro (fila).

```
DataRow dr=dt.Rows.Find(txtbuscar.Text);
    if (dr != null)
    {
        dr.Delete()
}
```

Actualizar un registro en un DataTable

Debemos localizar el registro a modificar y una vez modificado cambiar el valor de uno o varios de sus campos con algun valor pasado en algun control del formulario.

```
DataRow dr=dt.Rows.Find(txtbuscar.Text);
    if (dr != null)
    {
            dr["campo1"] = txtcontrol1.Text;
            dr["campo2"] = txtcontrol2.Text;
            ….

}
```

Terminar el ejemplo, intentando añadir un Editor (Publisher), borrar un editor y modificar los datos de un editor.

Si termináis, haced el ejercicio de productos trabajando en modo desconectado.