

Unidad 4 – Laboratorio 3

DataTable's

Objetivos

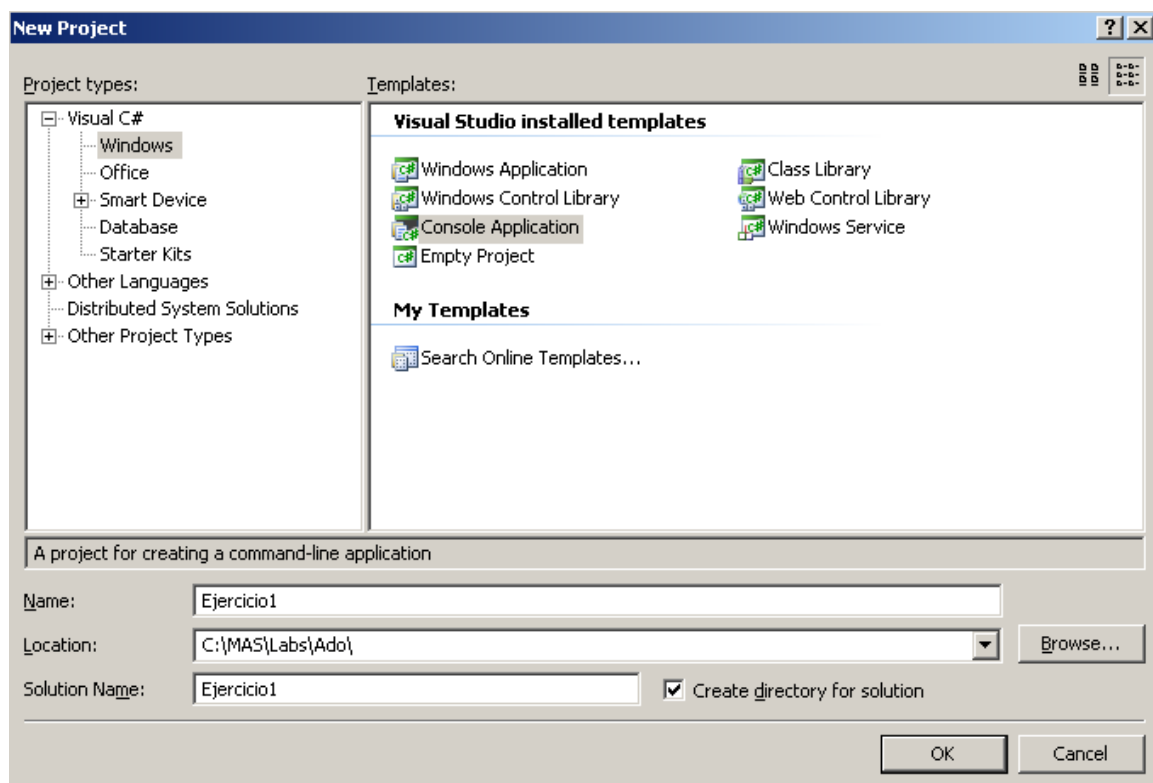
Al final de éste ejercicio usted tendrá la habilidad para realizar las operaciones básicas sobre un objeto DataTable y sus relaciones entre las diferentes fuentes de datos.

Duración Aproximada

35 minutos

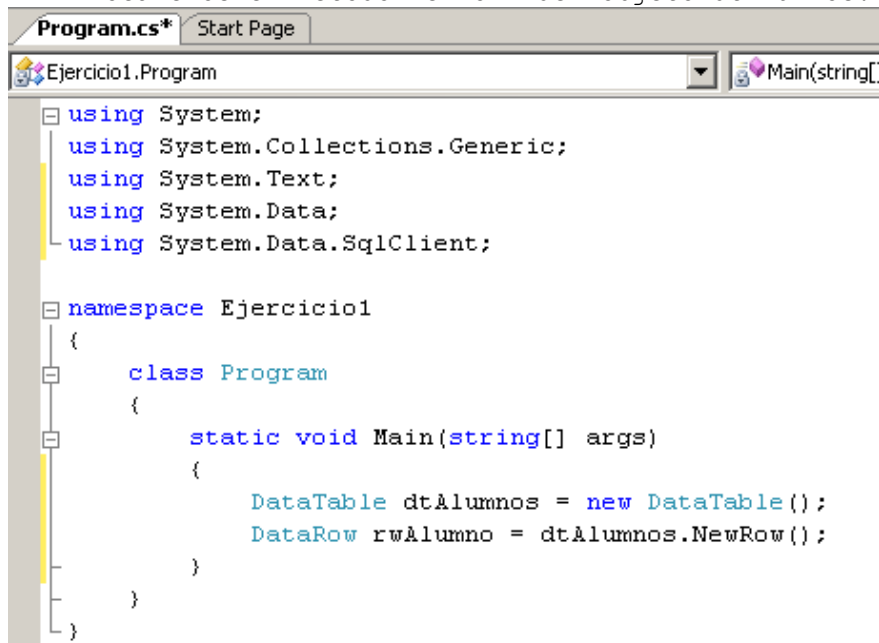
Pasos o Recomendaciones

- 1) Abra Visual Studio y cree en C# una aplicación de Consola, para ello seleccione la opción de menú File - New - Project



- 2) Seleccione el lenguaje de proyecto que usted más desea (por convención, todos los labs están escritos en C#)
- 3) Seleccione el template "Console Application"
- 4) En el cuadro de texto de "Name" escriba: Ejercicio1
- 5) Presione Aceptar
- 6) Agregue la directiva using (c#) en el archivo "Program.cs o .vb" para importar los namespace's de "system.data" y "system.data.sqlclient".

- 7) Dentro del método main declararemos un objeto del tipo DataTable llamado dtAlumnos
- 8) Luego crearemos otro objeto del tipo DataRow llamado rwAlumno e invocaremos el método NewRow del objeto dtAlumnos.



```
Program.cs* Start Page
Ejercicio1.Program
Main(string[] args)

using System;
using System.Collections.Generic;
using System.Text;
using System.Data;
using System.Data.SqlClient;

namespace Ejercicio1
{
    class Program
    {
        static void Main(string[] args)
        {
            DataTable dtAlumnos = new DataTable();
            DataRow rwAlumno = dtAlumnos.NewRow();
        }
    }
}
```

- 9) En siguiente paso crearemos dos objetos del tipo DataColumn, llamados dtcolNombre y dtcolApellido,
- 10) Estos objetos se llamaran Nombre y Apellido respectivamente y contendrán el tipo de datos String
- 11) Luego ingresaremos éstos dos objetos tipo DataColumn al objeto que hemos creado dtAlumnos
- 12) Siguiente paso será empezar a asignar valores al rwAlumno para insertarlo en nuestro DataTable.
- 13) Para ello escribiremos de la siguiente manera: tal como se muestra en la siguiente figura:

```
dtAlumnos.Columns.Add(colApellido);
dtAlumnos.Columns.Add(colNombre);

rwAlumno[colApellido] = "Perez";
rwAlumno[colNombre] = "Juan";

dtAlumnos.Rows.Add(rwAlumno);

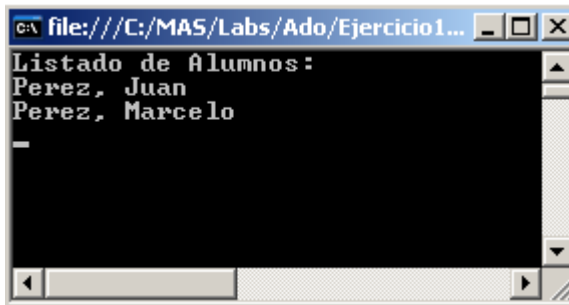
DataRow rwAlumno2 = dtAlumnos.NewRow();
rwAlumno2["Apellido"] = "Perez";
rwAlumno2["Nombre"] = "Marcelo";
dtAlumnos.Rows.Add(rwAlumno2);
```

- 14) Lo siguiente será volver a crear un objeto DataRow para volver agregar otro row a la tabla, y lo haremos como se representa en la figura anterior.

- 15) Luego procederemos a recorrer la colección de rows contenidos dentro del datatable
- 16) Antes vamos a escribir una línea en la consola de la aplicación y pondremos `Console.WriteLine("Listado de Alumnos")`
- 17) Ahora recorreremos la colección de datarows del datatable utilizando el `foreach(datarow row in dtAlumnos)`
- 18) Por cada row que recorremos lo mostramos en la consola, mostrando el apellido, seguido por una `' , '` (coma) y luego con el nombre
- 19) Por último escribiremos `Console.ReadLine()` para poder situar el cursor y visualizar los resultados.

```
Console.WriteLine("Listado de Alumnos:");  
foreach (DataRow row in dtAlumnos.Rows)  
{  
    Console.WriteLine(row[colApellido].ToString() + ", " + row[colNombre].ToString());  
}  
Console.ReadLine();
```

- 20) El resultado final será el siguiente:



DataSet's no tipados

Duración Aproximada

35 minutos

Pasos o Recomendaciones

Basándonos en el ejercicio anterior vamos a:

- 1) Eliminar toda las líneas que utilizamos para recorrer nuestro datatable y representar en pantalla los registros.

```
//-----  
Console.WriteLine("Listado de Alumnos:");  
foreach (DataRow row in dtAlumnos.Rows)  
{  
    Console.WriteLine(row[colApellido].ToString() + ", " + row[colNombre].ToString());  
}  
Console.ReadLine();
```

- 2) Luego vamos a agregar un DataColumn a nuestra tabla dtAlumnos del tipo int y que nos servirá como PrimaryKey. En la siguiente figura se muestra y donde hay que agregarlo, es importante destacar que el objeto datarow que nosotros declaramos anteriormente se debe declarar luego de agregarles los datacolumns a la tabla, por lo que hay que mover esa línea más abajo:

```
{  
    DataTable dtAlumnos = new DataTable();  
  
    DataColumn colNombre = new DataColumn("Nombre", typeof(string));  
    DataColumn colApellido = new DataColumn("Apellido", typeof(string));  
    dtAlumnos.Columns.Add(colApellido);  
    dtAlumnos.Columns.Add(colNombre);  
    DataRow rwAlumno = dtAlumnos.NewRow();  
    rwAlumno[colApellido] = "Perez";  
    rwAlumno[colNombre] = "Juan";  
  
    dtAlumnos.Rows.Add(rwAlumno);  
  
    DataRow rwAlumno2 = dtAlumnos.NewRow();  
    rwAlumno2["Apellido"] = "Perez";  
    rwAlumno2["Nombre"] = "Marcelo";  
    dtAlumnos.Rows.Add(rwAlumno2);  
}
```

- 3) Ahora agregamos el nuevo datacolumn llamado colIDAlumno, del tipo int y le establecemos el autoncrement = true, y luego lo agregamos a nuestra tabla:

```

    DataTable dtAlumnos = new DataTable();
→ DataColumn colIDAlumno = new DataColumn("IDAlumno", typeof(int));
→ colIDAlumno.ReadOnly = true; //Indicamos que será readonly
→ colIDAlumno.AutoIncrement = true; //Será autoincremental
→ colIDAlumno.AutoIncrementSeed = 0; //El primer número será 0
→ colIDAlumno.AutoIncrementStep = 1; //se incrementará de a 1
    DataColumn colNombre = new DataColumn("Nombre", typeof(string));
    DataColumn colApellido = new DataColumn("Apellido", typeof(string));
    dtAlumnos.Columns.Add(colApellido);
    dtAlumnos.Columns.Add(colNombre);
→ dtAlumnos.Columns.Add(colIDAlumno); //lo agregamos al datatable
    DataRow rwAlumno = dtAlumnos.NewRow();
    rwAlumno[colApellido] = "Perez";
    rwAlumno[colNombre] = "Juan";

```

- 4) Ahora vamos a establecer ése campo como PrimaryKey de la tabla dtAlumnos de ls siguiente manera:

```

    DataColumn colIDAlumno = new DataColumn("IDAlumno", typeof(int));
    colIDAlumno.ReadOnly = true;
    colIDAlumno.AutoIncrement = true;
    colIDAlumno.AutoIncrementSeed = 0;
    colIDAlumno.AutoIncrementStep = 1;
    DataColumn colNombre = new DataColumn("Nombre", typeof(string));
    DataColumn colApellido = new DataColumn("Apellido", typeof(string));
    dtAlumnos.Columns.Add(colApellido);
    dtAlumnos.Columns.Add(colNombre);
    dtAlumnos.Columns.Add(colIDAlumno);
    //Establecemos la clave primaria
→ dtAlumnos.PrimaryKey = new DataColumn[] { colIDAlumno };
    DataRow rwAlumno = dtAlumnos.NewRow();
    rwAlumno[colApellido] = "Perez";
    rwAlumno[colNombre] = "Juan";

```

- 5) Por consiguiente, a partir de ahora en más cada ves que agregemos un datarow al datatable indicandole el nombre y el apellido se autoincrementará el valor en el campo IDAlumno. Visualizandose de la siguiente manera (usando el QuickWatch)

DataSet Visualizer

Table:

	Apellido	Nombre	IDAlumno
▶	Perez	Juan	0
	Perez	Marcelo	1

- 6) Ahora el siguiente paso será crear otra DataTable llamado dtCursos y que contendrá los DataColumn de: colIDCurso (tipo int, incremental y PrimaryKey) y el datacolum colCurso. Todo esto

lo realizaremos debajo de toda nuestra linea de código, tal como se muestra en la figura siguiente:

```
rwAlumno2["Apellido"] = "Perez";
rwAlumno2["Nombre"] = "Marcelo";
dtAlumnos.Rows.Add(rwAlumno2);

DataTable dtCursos = new DataTable("Cursos");
DataColumn colIDCurso = new DataColumn("IDCurso", typeof(int));
colIDCurso.ReadOnly = true;
colIDCurso.AutoIncrement = true;
colIDCurso.AutoIncrementSeed = 1;
colIDCurso.AutoIncrementStep = 1;
DataColumn colCurso = new DataColumn("Curso", typeof(string));
dtCursos.Columns.Add(colIDCurso);
dtCursos.Columns.Add(colCurso);
dtCursos.PrimaryKey = new DataColumn[] { colIDCurso };
```

- 7) Agregamos 1 datarow a dicha tabla con el nombre de curso:
"Informatica"

```
DataRow rwCurso = dtCursos.NewRow();
rwCurso[colCurso] = "Informatica";
dtCursos.Rows.Add(rwCurso);
```

- 8) Ahora que tenemos éstos dos datatables llamados dtAlumnos y dtCursos lo que haremos es incorporarlos dentro de un objeto DataSet llamado dsUniversidad

```
DataSet dsUniversidad = new DataSet();
dsUniversidad.Tables.Add(dtAlumnos);
dsUniversidad.Tables.Add(dtCursos);
```

- 9) Lo que nos falta a éste ejercicio es crear una tabla que actue de relacion entre el alumno y los cursos, y como la relacion es de muchos a muchos, debemos agregar una tabla intermedia entre ámbas que la llamaremos dtAlumnos_Cursos y que contendrá los datacolumns col_ac_IDAlumno del tipo int y el datacolumn col_ac_IDCurso

```
DataTable dtAlumnos_Cursos = new DataTable("Alumnos_Cursos");
DataColumn col_ac_IDAlumno = new DataColumn("IDAlumno", typeof(int));
DataColumn col_ac_IDCurso = new DataColumn("IDCurso", typeof(int));
dtAlumnos_Cursos.Columns.Add(col_ac_IDAlumno);
dtAlumnos_Cursos.Columns.Add(col_ac_IDCurso);

dsUniversidad.Tables.Add(dtAlumnos_Cursos);
```

- 10) Ahora tenemos 3 tablas dentro del dataset dsUniversidad, del cual dos de aquellas tablas (alumnos y cursos tienen registros), solo falta relacionarlas entre ellas. Por lo que primeramente crearemos un objeto DataRelation entre la tabla dtAlumnos y dtAlumnos_Cursos de la siguiente manera:

```

DataRelation relAlumno_ac =
    new DataRelation("Alumno_Cursos", colIDAlumno, col_ac_IDAlumno);
DataRelation relCurso_ac =
    new DataRelation("Curso_Alumnos", colIDCurso, col_ac_IDCurso);
dsUniversidad.Relations.Add(relAlumno_ac);
dsUniversidad.Relations.Add(relCurso_ac);

```

11) En la figura anterior también hemos agregado la relación entre la tabla Cursos y la tabla intermedia Alumnos_Cursos

12) Hasta aquí hemos visto como crear objetos DataTable, DataColumn, DataRow, DataSet y DataRelation.

13) En siguiente lugar vamos a crear los registros dentro de la tabla intermedia para asociar un alumno a un curso. Para ello lo que haremos es (sabiendo de antemano que el IDAlumno de "Juan Perez" es 0 y el IDCurso de "Informatica" es 1, agregaremos un registro en la tabla dtAlumnos_Cursos con esos valores, de la siguiente manera:

```

DataRow rwAlumnosCursos = dtAlumnos_Cursos.NewRow();
rwAlumnosCursos[col_ac_IDAlumno] = 0; //Alumno: Juan Perez
rwAlumnosCursos[col_ac_IDCurso] = 1; //Curso: Informatica
dtAlumnos_Cursos.Rows.Add(rwAlumnosCursos);

rwAlumnosCursos = dtAlumnos_Cursos.NewRow();
rwAlumnosCursos[col_ac_IDAlumno] = 1; //Alumno: Marcelo Perez
rwAlumnosCursos[col_ac_IDCurso] = 1; //Curso: Informatica
dtAlumnos_Cursos.Rows.Add(rwAlumnosCursos);

```

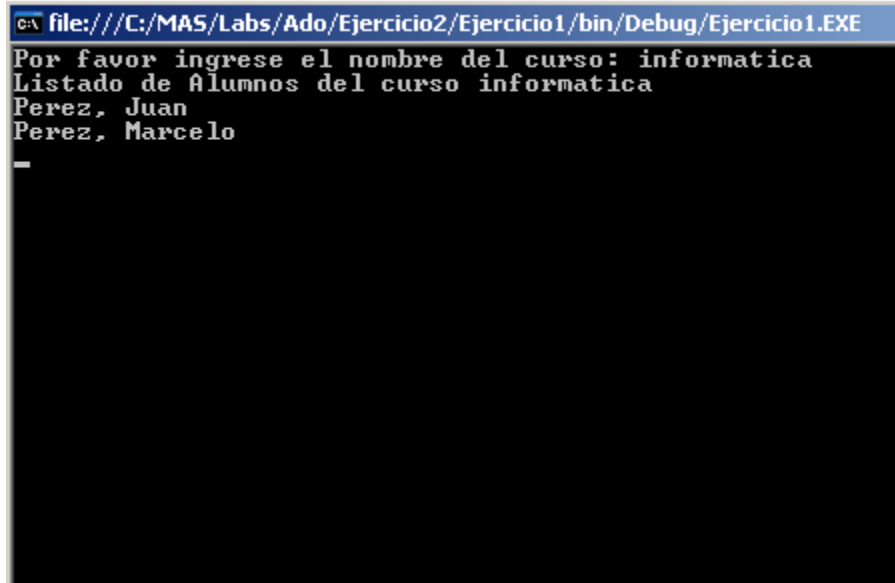
14) Como último paso nos quedaria recorrer esos registros, pero en éste caso vamos a mostrar en consola todos los alumnos que estan asociados al curso que indicaremos previamente, para ello procederemos de ls siguiente manera

```

Console.WriteLine("Por favor ingrese el nombre del curso: ");
string materia = Console.ReadLine();
Console.WriteLine("Listado de Alumnos del curso " + materia);
DataRow[] row_CursoInf = dtCursos.Select("Curso = '" + materia + "'");
foreach (DataRow rowCu in row_CursoInf)
{
    DataRow[] row_AlumnosInf = rowCu.GetChildRows(relCurso_ac);
    foreach (DataRow rowAl in row_AlumnosInf)
    {
        Console.WriteLine(
            rowAl.GetParentRow(relAlumno_ac)[colApellido].ToString()
            + ", " +
            rowAl.GetParentRow(relAlumno_ac)[colNombre].ToString()
        );
    }
}
Console.ReadLine();

```

15) El resultado final obtenido será el siguiente:



A screenshot of a Windows command prompt window. The title bar at the top reads "file:///C:/MAS/Labs/Ado/Ejercicio2/Ejercicio1/bin/Debug/Ejercicio1.EXE". The command prompt shows the following text: "Por favor ingrese el nombre del curso: informatica", "Listado de Alumnos del curso informatica", "Perez, Juan", "Perez, Marcelo", and a single hyphen "-" on the next line. The background of the command prompt is black, and the text is white.

16) Hasta aquí hemos visto como utilizar básicamente los objetos y por ende consumirlos

16) Fin del Ejercicio

DataSet's Tipados

Objetivos

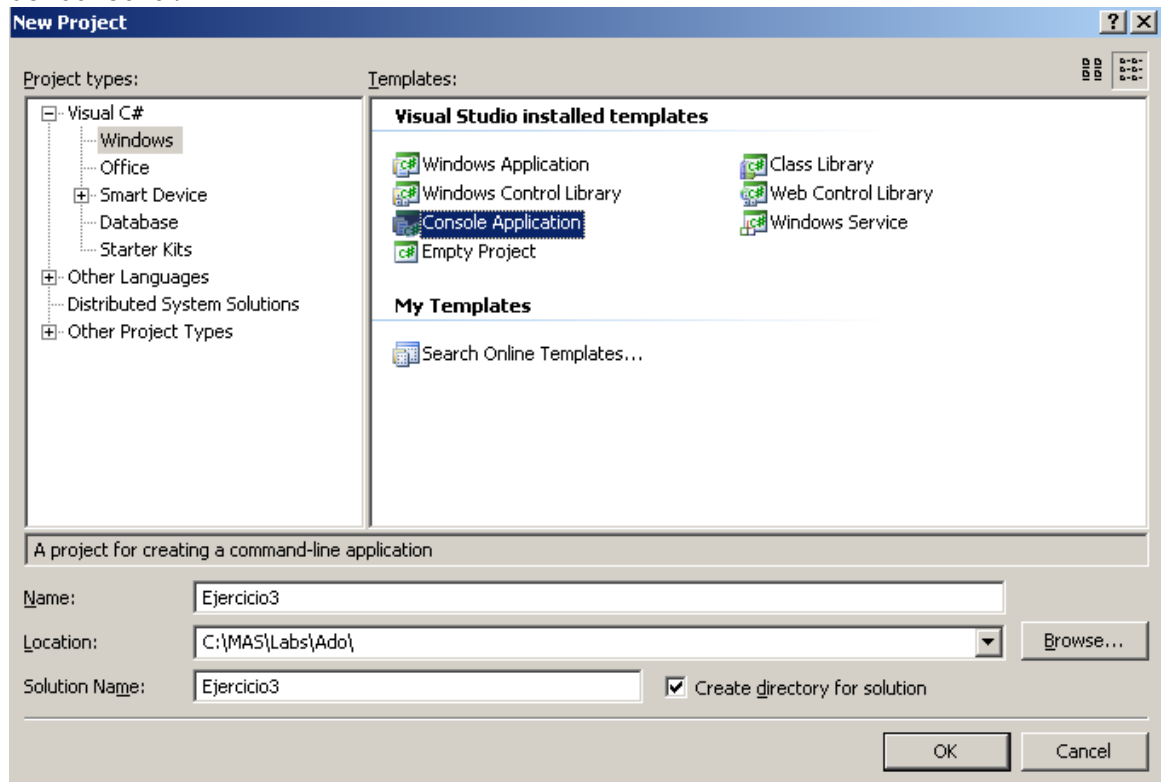
En base de lo realizado hasta el momento, procederemos a realizar otro dataset en forma tipada, y veremos las diferencias y las ventajas que éste ofrece.

Duración Aproximada

30 minutos

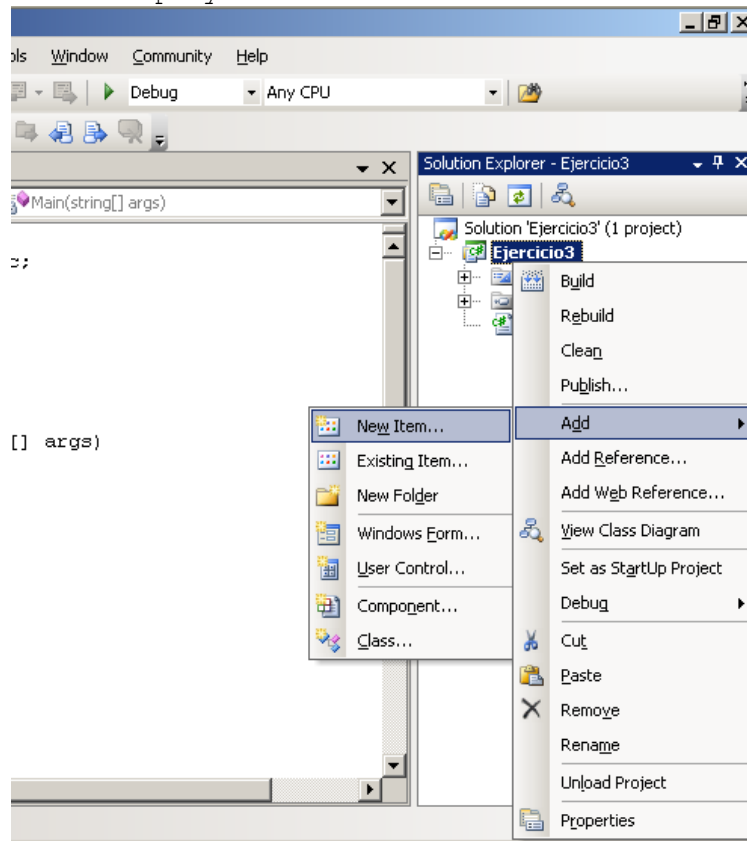
Pasos:

- 1) En base a lo realizado en el ejercicio anterior procederemos a volver a realizarlo pero viendo las características y beneficios en la utilización de los DataSet's Tipados
- 2) Para ello crearemos un nuevo proyecto llamado Ejercicio3 del tipo de Consola:

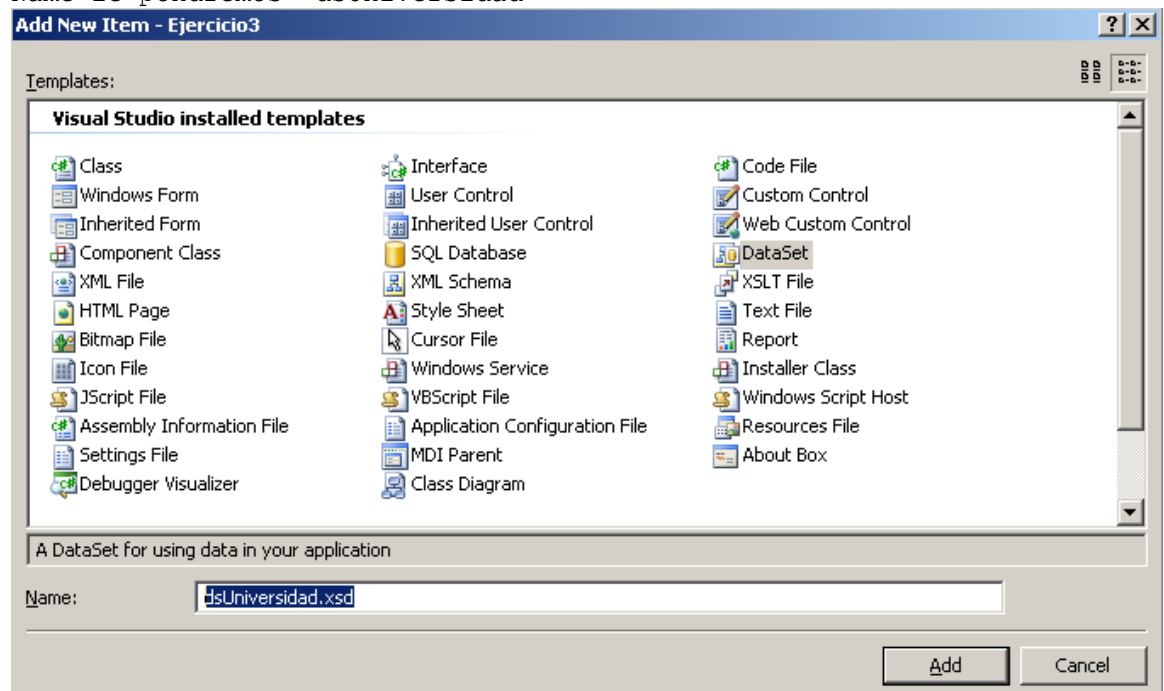


- 3) Abriremos el Solution Explorer ubicado en la parte derecha del Visual Studio, o bien desde el menú View-Solution Explorer

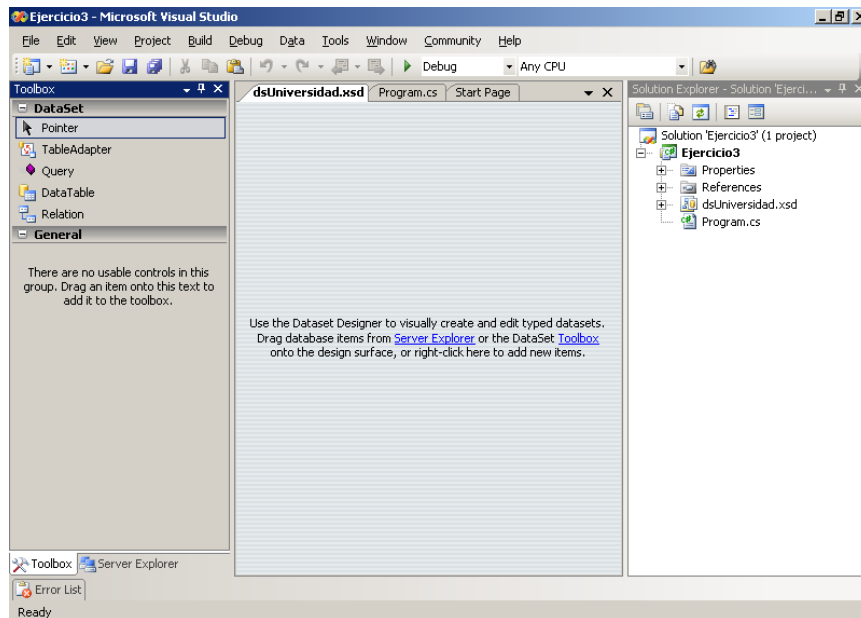
- 4) Agregaremos un Nuevo elemento al proyecto haciendo click derecho sobre el proyecto:



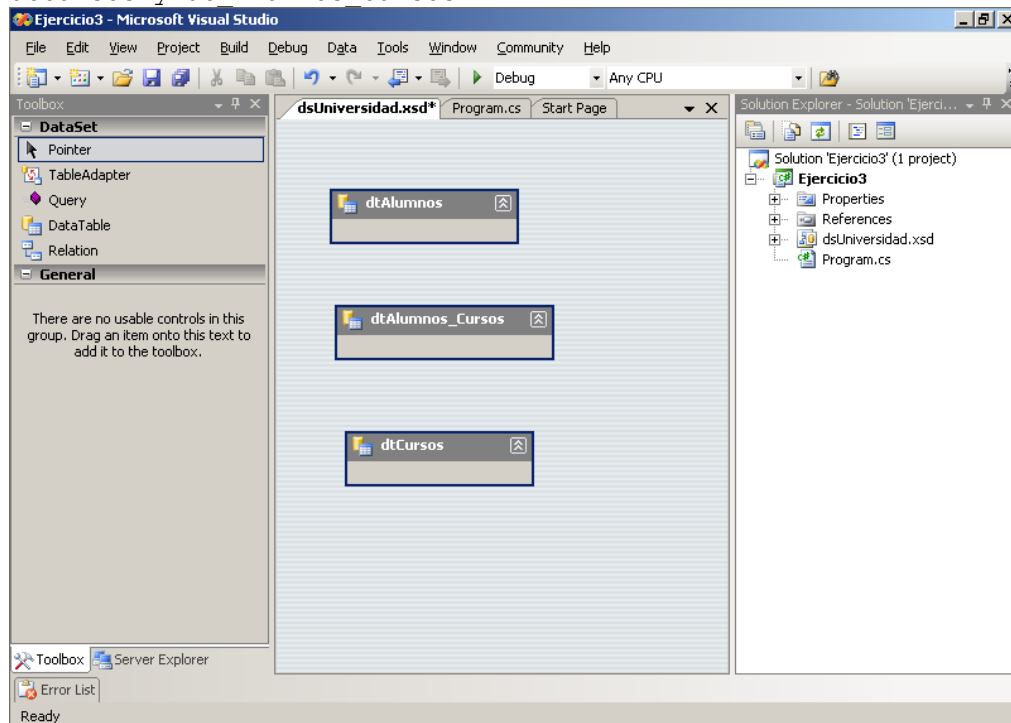
- 5) Y en la lista de templates seleccionaremos el item DataSet como Name le pondremos "dsUniversidad"



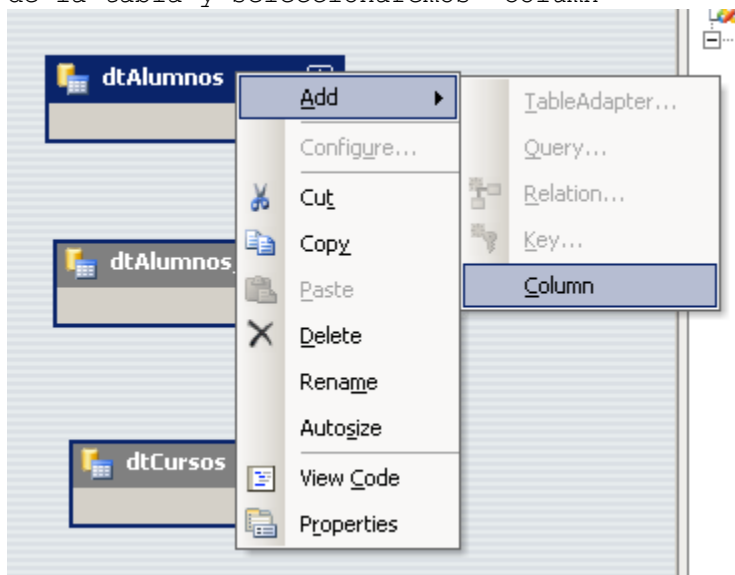
- 6) Presionamos "Add" y dentro de nuestro proyecto aparecerá un file llamado dsUniversidad.xsd (esquema XML) como se muestra en la siguiente figura:



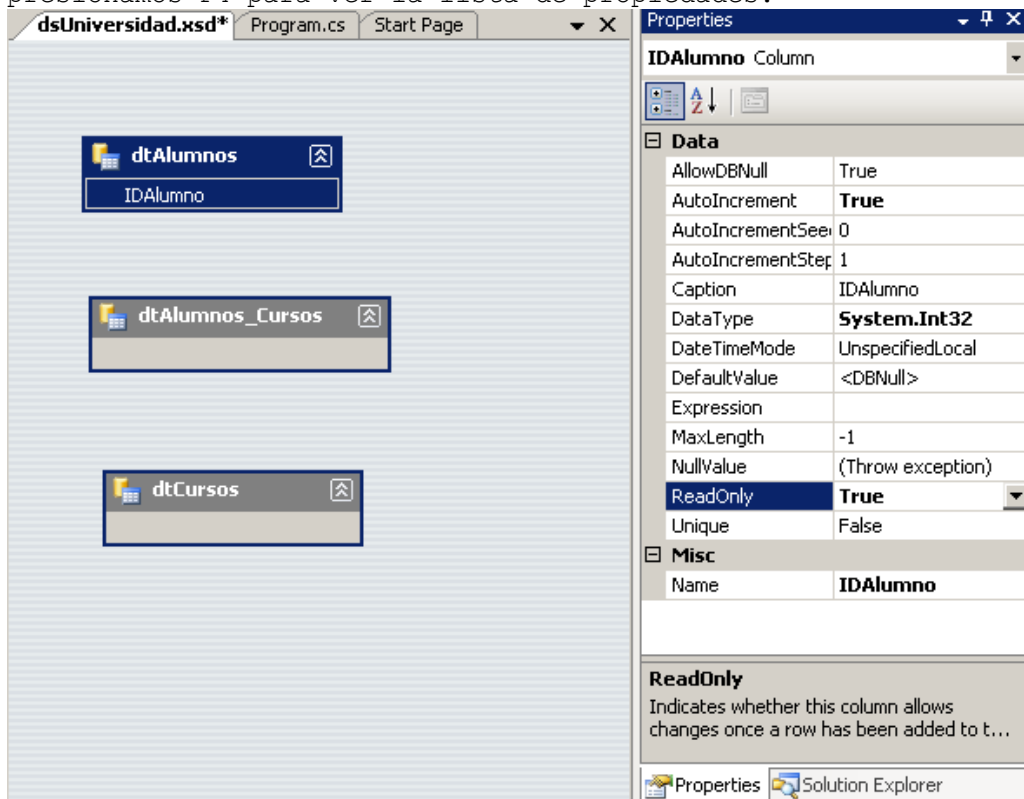
- 7) Ahora que hemos agregado un esquema xml a nuestro proyecto iremos a nuestro ToolBox ubicado en el extremo izquierdo de Visual Studio y seleccionaremos el objeto DataTable y lo arrastraremos hacia la superficie del dataset
- 8) Seleccionamos el titulo (que por defecto se llama DataTable1) y le pondremos el nombre de dtAlumnos
- 9) De la misma forma procedemos a agregar 2 datatables más llamados dtCursos y dt Alumnos Cursos



- 10) Ahora que hemos creado los 3 datatables, vamos a agregar a cada uno de ellos los datacolumns correspondientes, primeramente sobre la tabla dtAlumnos vamos hacer click derecho sobre el título de la tabla y seleccionaremos "Column"



- 11) Una vez que agregamos el datacolumn a la tabla, le ponemos el nombre de IDAlumno y seleccionamos el column correspondiente y presionamos F4 para ver la lista de propiedades.

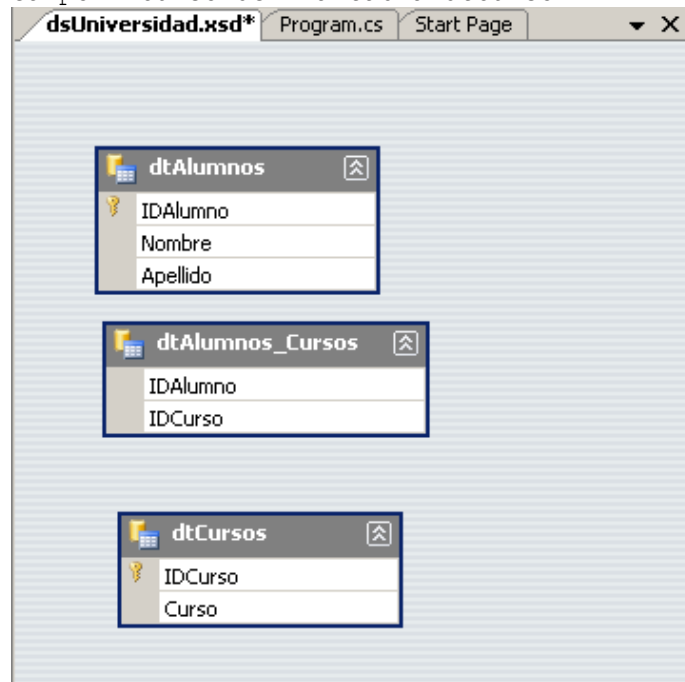


- 12) De la lista de propiedades, le indicamos AutoIncrement = true, DataType = Int32 y ReadOnly = true

- 13) De igual manera procederemos a agregar todos los datacolumn en las tablas con las siguiente propiedades:

DataTable	DataColumn	Properties
dtAlumnos	IDAlumno	Datatype = Int32
		AutoIncrement = 0
		ReadOnly = true
	Nombre	Datatype = string
dtCursos	IDCurso	Datatype = Int32
		AutoIncrement = 1
		ReadOnly = true
	Curso	Datatype = string
dtAlumnos_Cursos	IDAlumno	Datatype = Int32
	IDCurso	Datatype = Int32

- 14) Una vez creado todos los campos de las tablas agregaremos indicaremos las Claves Primarias a la tabla dtAlumnos sobre el campo IDAlumno, y la tabla dtCursos sobre el campo IDCurso
- 15) Para establecer un campo como clave primaria (PrimaryKey) de una tabla simplemente seleccionamos el datacolumn y realizamos click derecho sobre el margen izquierdo del datacolumn y seleccionamos "Set Primary Key". Lo mismo realizamos sobre el campo IDCurso del la tabla dtCurso



- 16) Luego de tener las 3 tablas dentro del dataset, vamos a agregar la relacion entre ellas. Seleccionamos del ToolBox DataRelation, lo arrastramos y lo tiramos sobre la superficie del dataset

Relation

Name:

Specify the keys that relate tables in your dataset.

Parent Table: Child Table:

Columns:

Key Columns	Foreign Key Columns
IDAlumno	IDAlumno

Choose what to create

☐ Both Relation and Foreign Key Constraint
☐ Foreign Key Constraint Only
☒ Relation Only

Update Rule:

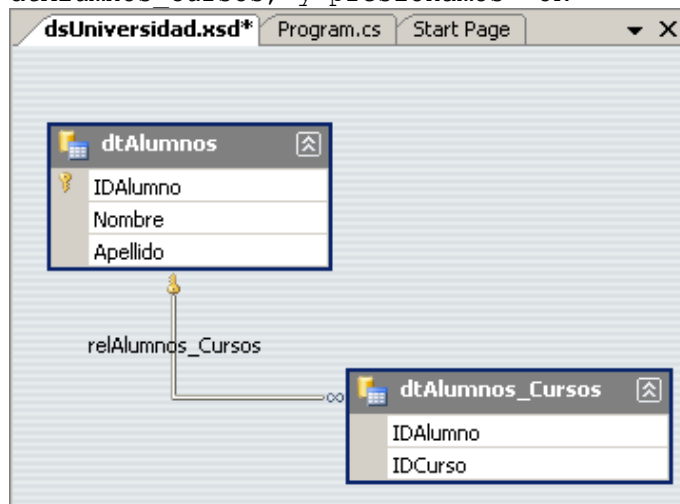
Delete Rule:

Accept/Reject Rule:

☐ Nested Relation

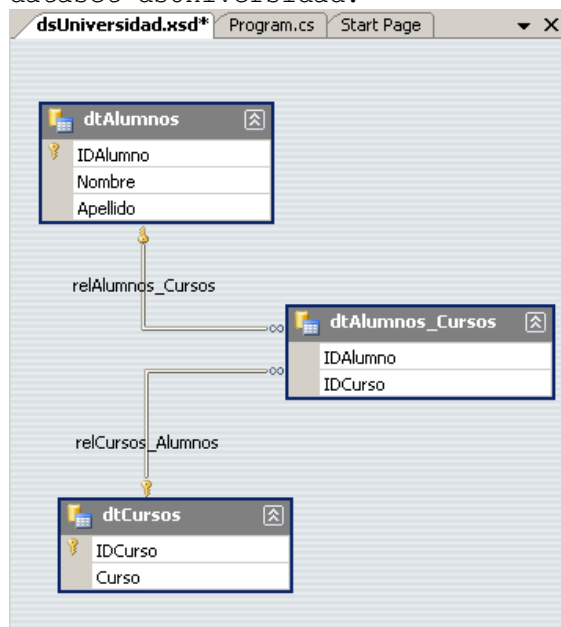
OK Cancel

- 17) En la figura anterior cambiamos en el textbox Name, le ponemos el nombre de "relAlumnos_Cursos" y como ParentTable seleccionamos la tabla dtAlumnos; como ChildTable, seleccionamos dtAlumnos Cursos, y presionamos "OK"



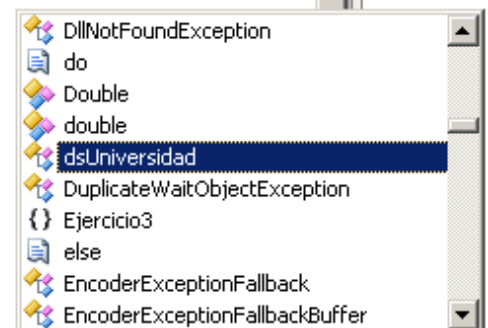
- 18) Si deseamos visualizarlo de una mejor manera, podemos seleccionar el encabezado del datatable y desplazarlo dentro del dataset, para una mejor visualización

- 19) De la misma manera lo vamos a agregar un objeto DataRelation entre la tabla dtCursos y dtAlumnos_Cursos y lo llamaremos "relCursos_Alumnos" quedando definitivamente así el dataset dsUniversidad:



- 20) Hasta aquí, hemos aprendido a crear un DataSet Tipado, viendo notado la sencillez y el tiempo en que se diseña, vamos a ver ahora las ventajas que esto nos conlleva en el uso del mismo:
- 21) Antes que nada presionaremos F6 o "Build" desde el menú Debug Build para compilar la aplicación
- 22) Ahora seleccionaremos desde la solapa de documentos abiertos el documento Program.cs
- 23) Dentro del método main crearemos un objeto llamado miUniversidad del tipo dsUniversidad, notese que en la lista de miembros aparece dsUniversidad

```
namespace Ejercicio3
{
    class Program
    {
        static void Main(string[] args)
        {
            dsUniversidad miUniversidad = new
        }
    }
}
```



Propiedades

- 24) Luego crearemos un objeto por cada DataTable contenido dentro de miUniversidad de la siguiente manera:
- ```
dsUniversidad miUniversidad = new dsUniversidad();

dsUniversidad.dtAlumnosDataTable dtAlumnos
 = new dsUniversidad.dtAlumnosDataTable();
dsUniversidad.dtCursosDataTable dtCursos
 = new dsUniversidad.dtCursosDataTable();
dsUniversidad.dtAlumnosRow rowAlumno = dtAlumnos.NewdtAlumnosRow();

rowAlumno.Apellido = "Perez";
rowAlumno.Nombre = "Juan";
dtAlumnos.AdddtAlumnosRow(rowAlumno);
```
- 25) Notese en la figura anterior que declaramos los objetos dtAlumnos y dtCursos y hemos agregado dos registros a la tabla de alumnos. Eh aquí una gran diferencia entre la forma tipada y no tipada. En éste tipo de objetos, tenemos la ventaja de que el DataSet designer nos crea una clase y expone dentro de esa clase los miembros necesarios para trabajar.
- 26) Resta tambien agregar un curso a la tabla de dtCursos, y agregaremos (tal como lo vinimos haciendo en el ejercicio anterior), agregaremos el curso de "Informatica"
- ```
dsUniversidad.dtCursosRow rowCurso = dtCursos.NewdtCursosRow();
rowCurso.Curso = "Informatica";
dtCursos.AdddtCursosRow(rowCurso);
```
- 27) ahora crearemos la relacion entre el alumno Juan Perez y el curso Informatica:
- ```
//Primero creamos el objeto datatable
dsUniversidad.dtAlumnos_CursosDataTable dtAlumnos_Cursos
 = new dsUniversidad.dtAlumnos_CursosDataTable();

dsUniversidad.dtAlumnos_CursosRow rowAlumnosCursos =
 dtAlumnos_Cursos.NewdtAlumnos_CursosRow();

dtAlumnos_Cursos.AdddtAlumnos_CursosRow(rowAlumno, rowCurso);
```
- 28) En éste ultimo vemos que se le debe indicar directamente el row del alumno y del curso, por lo que no necesitamos saber de antemano cual es el IDAlumno ni el IDCurso. Simplemente asociamos éstos dos row's.
- 29) Creado el curso "Informatica" y el alumno "Juan Perez" y asociados en la tabla dtAlumnos\_Cursos, vamos a crear el otro alumno llamado "Marcelo Perez":
- ```
rowAlumno = dtAlumnos.NewdtAlumnosRow();
rowAlumno.Nombre = "Marcelo";
rowAlumno.Apellido = "Perez";
dtAlumnos.AdddtAlumnosRow(rowAlumno);

dtAlumnos_Cursos.AdddtAlumnos_CursosRow(rowAlumno, rowCurso);
```


- 30) En éste punto hemos llegado a comprender y aplicar el uso de los dataset tipados, aprendimos a diseñarlos y a cargarlos con registros.
- 31) Fin del ejercicio.

Objeto SqlConnection y SqlDataReader

Objetivo

Aquí veremos como consumir datos desde una fuente de datos (en éste caso utilizaremos la base de datos de Northwind de SQL)

Duración Aproximada

20 minutos

Pasos o Recomendaciones

- 1) Primeramente veremos como crear una conexión con SQLServer utilizando el objeto SqlConnection ubicado en el namespace System.Data.SqlClient
- 2) Vamos a crear un nuevo proyecto llamado Ejercicio4 del tipo aplicación de consola
- 3) Primeramente vamos a importar los namespaces System.Data y System.Data.SqlClient
- 4) Luego dentro del método main declararemos un objeto llamado dtEmpresas del tipo DataTable y le agregaremos dos DataColumn llamados CustomerID y CompanyName

```
static void Main(string[] args)
{
    //Creo un objeto DataTable llamado Empresas
    DataTable dtEmpresas = new DataTable("Empresas");
    //Al objeto DataTable le agrego dos datacolumns del tipo string
    dtEmpresas.Columns.Add("CustomerID", typeof(string));
    dtEmpresas.Columns.Add("CompanyName", typeof(string));
```
- 5) Declaramos un objeto sqlconnection llamado myconn y le indicaremos el connectionstring que utilizará (antes de realizar la operación, se debe verificar que el connection string sea el correcto en el ámbito donde se va a ejecutar)

```
//Creamos un objeto SqlConnection
SqlConnection myconn = new SqlConnection();
//Indicamos el Connection String que utilizará
myconn.ConnectionString =
    "Data Source=LOCALHOST;Initial Catalog=Northwind;User ID=sa;Pwd=123";
```
- 6) Creamos un objeto SqlCommand y le indicamos la cadena TSQL que utilizará y el objeto sqlconnection que utilizará

```

//Creo un objeto SqlCommand
SqlCommand mycomando = new SqlCommand();
//Le indico la cadena TSQL que utilizará
mycomando.CommandText = "SELECT CustomerID, CompanyName FROM Customers";
//Indico el objeto connection que utilizará
mycomando.Connection = myconn;

```

- 7) Creamos ahora un objeto SqlDataReader y le indicamos el Select Command Text y el objeto sqlconnection que utilizará

```

//Creamos un adaptador del tipo SqlDataAdapter y
//Le indicamos el command text que utilizará para realizar la consulta
//y el objeto sqlconnection que utiliza
SqlDataAdapter myadap =
    new SqlDataAdapter("SELECT CustomerID, CompanyName FROM Customers", myconn);

```

- 8) Ahora abrimos la coneccion con la base de datos. Creamos un objeto DataReader llamado mydr e invocamos el método ExecuteReader del objeto mycomando que anteriormente hemos creado.

- 9) Luego cargamos el objeto dtEmpresas, utilizando el método Load y pasandole en el argumento el objeto Datareader. Por último cerramos la coneccion

```

//Abro la coneccion
myconn.Open();
//Creo un objeto DataReader y ejecuto el método
//ExecuteReader del objeto mycomando
SqlDataReader mydr = mycomando.ExecuteReader();
//Cargo los datos en el datatable utilizando el objeto DataReader
dtEmpresas.Load(mydr);
//Cierro la coneccion
myconn.Close();

```

- 10) Por último vamos a recorrer los registros obtenidos y representarlos en la consola

```

//Recorro la lista de empresas obtenidas
//y lo muestro en consola
Console.WriteLine("Listado de Empresas: ");
foreach (DataRow rowEmpresa in dtEmpresas.Rows)
{
    string idempresa = rowEmpresa["CustomerID"].ToString();
    string nombreempresa = rowEmpresa["CompanyName"].ToString();
    Console.WriteLine(idempresa + " - " + nombreempresa);
}
Console.ReadLine();

```

- 11) El resultado final será el siguiente:

```
file:///C:/MAS/Labs/Ado/Ejercicio4/Ejercicio4/bin/Debug/Ejercicio4.EXE
Listado de Empresas:
ALFKI - Alfreds Futterkiste
ANATR - Ana Trujillo Emparedados y helados
ANTON - Antonio Moreno Taquería
AROUT - Around the Horn
BERGS - Berglunds snabbköp
BLAUS - Blauer See Delikatessen
BLONP - Blondesdds1 père et fils
BOLID - Bólido Comidas preparadas
BONAP - Bon app'
BOTTM - Bottom-Dollar Markets
BSBEV - B's Beverages
CACTU - Cactus Comidas para llevar
CENTC - Centro comercial Moctezuma
CHOPS - Chop-suey Chinese
COMMI - Comércio Mineiro
CONSH - Consolidated Holdings
WANDK - Die Wandernde Kuh
DRACD - Drachenblut Delikatessen
DUMON - Du monde entier
EASTC - Eastern Connection
ERNSH - Ernst Handel
FAMIA - Familia Arquibaldo
FISSA - FISSA Fábrica Inter. Salchichas S.A.
FOLIG - Folies gourmandes
```

Hasta aquí, hemos aprendido a llenar un datatable por medio de su método Load y indicándole un objeto DataReader.

Objeto SqlConnection y SqlDataAdapter (.Fill)

Objetivo

Obtener conocimientos básicos en el manejo de los objetos connection y el objeto dataadapter

Duración Aproximada

15 minutos

Pasos o Recomendaciones

- 1) Basado en el ejercicio anterior (de la clase pasada) , ahora veremos de realizar la misma operación pero utilizando el objeto SqlDataAdapter en ves del objeto DatarReader.
- 2) Lo que vamos a hacer primeramente es eliminar el código que se muestra encuadrado a continuación

```
//Creamos un objeto SqlConnection
SqlConnection myconn = new SqlConnection();
//Indicamos el Connection String que utilizará
myconn.ConnectionString =
    "Data Source=BEYONDO7;Initial Catalog=Northwind;User ID=sa;Pwd=123";
```

```
//Creo un objeto SqlCommand
SqlCommand mycomando = new SqlCommand();
//Le indico la cadena TSQL que utilizará
mycomando.CommandText = "SELECT CustomerID, CompanyName FROM Customers";
//Indico el objeto connection que utilizará
mycomando.Connection = myconn;
```

```
//Abro la coneccion
myconn.Open();
//Creo un objeto DataReader y ejecuto el método
//ExecuteReader del objeto mycomando
SqlDataReader mydr = mycomando.ExecuteReader();
//Cargo los datos en el datatable utilizando el objeto DataReader
dtEmpresas.Load(mydr);
//Cierro la coneccion
myconn.Close();
```

```
//Recorro la lista de empresas obtenidas
//y lo muestro en consola
Console.WriteLine("Listado de Empresas: ");
```

- 3) En el lugar que previamente borramos creamos un objeto SqlDataAdapter y le indicamos el Select Command Text y el objeto sqlconnection que utilizará

```

//Creamos un adaptador del tipo SqlDataAdapter y
//Le indicamos el command text que utilizará para realizar la consulta
//y el objeto sqlconnection que utiliza
SqlDataAdapter myadap =
    new SqlDataAdapter("SELECT CustomerID, CompanyName FROM Customers", myconn);

```

- 4) Ahora abrimos la conexión con la base de datos y rellenamos el contenido obtenido en el objeto DataTable llamado dtEmpresas. Por último cerramos la conexión

```

//Abro la conexión
myconn.Open();
//Cargo el contenido del result set obtenido de la base
//de datos en el objeto datatable
myadap.Fill(dtEmpresas);
//Cierro la conexión
myconn.Close();

```

Por último ejecutamos la aplicación presionando F5 y vemos los mismos resultados que antes



```

file:///C:/MAS/Labs/Ado/Ejercicio4/Ejercicio4/bin/Debug/Ejercicio4.EXE
Listado de Empresas:
ALFKI - Alfreds Futterkiste
ANATR - Ana Trujillo Emparedados y helados
ANTON - Antonio Moreno Taquería
AROUT - Around the Horn
BERGS - Berglunds snabbköp
BLAUS - Blauer See Delikatessen
BLONP - Blondesdds1 père et fils
BOLID - Bólido Comidas preparadas
BONAP - Bon app'
BOTTM - Bottom-Dollar Markets
BSBEU - B's Beverages
CACTU - Cactus Comidas para llevar
CENTC - Centro comercial Moctezuma
CHOPS - Chop-suey Chinese
COMMI - Comércio Mineiro
CONSH - Consolidated Holdings
WANDK - Die Wandernde Kuh
DRACD - Drachenblut Delikatessen
DUMON - Du monde entier
EASTC - Eastern Connection
ERNSH - Ernst Handel
FAMIA - Familia Arquibaldo
FISSA - FISSA Fabrica Inter. Salchichas S.A.
FOLIG - Folies gourmandes

```

- 5) Hasta aquí, hemos aprendido a llenar utilizando el método fill del objeto dataadapter para llenar el contenido en un datatable

Objeto SqlConnection y SqlDataAdapter (.Update)

Objetivo

Obtener conocimientos básicos en el manejo de los objetos connection y el objeto dataadapter utilizando el método Update()

Duración Aproximada

20 minutos

Pasos o Recomendaciones

- 1) Basado en el ejercicio anterior, ahora veremos de realizar la misma operación pero vamos a realizar cambios en nuestro datatable y luego volcarlos a la base de datos. De ésta manera pondremos en práctica el modelo de acceso a datos desconectados.
- 2) Utilizando el mismo ejercicio anterior vamos a ubicarnos al final del código y vamos a agregar el siguiente código.

```
//Primero indico el CustomerID que deseo modificar
Console.Write("Escriba el CustomerID que desea modificar: ");
string custid = Console.ReadLine();
```

- 3) Ahora recorremos la colección de datarows en busca del customerid

```
//Luego me traigo una coleccion de datarows que contengan ese customerid
DataRow[] rwempresas = dtEmpresas.Select("CustomerID = '" + custid + "'");
if (rwempresas.Length != 1) //Si no encuentro nada entonces salgo
{
    Console.WriteLine("CustomerID no encontrado");
    Console.ReadLine();
    return;
}
```
- 4) Si no llegamos a encontrar el customerid entonces enviamos un mensaje y finalizamos la ejecucion del programa
- 5) Si encontramos el customerid, entonces me traigo el DataRow correspondiente que tiene dicho customerid y muestro en consola el nombre original y solicito un nuevo nombre para dicha empresa:

```
//Me traigo el primer datarow de la colección
DataRow rowMiEmpresa = rwempresas[0];
string nombreactual = rowMiEmpresa["CompanyName"].ToString();
//Muestro en consola el nombre del customerid seleccionado
Console.WriteLine("Nombre actual de la empresa: " + nombreactual);
//Solicito que escriba un nuevo nombre
Console.Write("Escriba el nuevo nombre: ");
string nuevonombre = Console.ReadLine();
```

- 6) Ahora que eh obtenido el nuevo nombre, modifico el datarow, pero primeramente hay que llamar al método .BeginEdit del propio

datarow para luego poder asignarles valores y por último se debe llamar el método EndEdit() como se muestra en la siguiente figura:

```
//Llamo al método BeginEdit del datarow para iniciar los cambios
rowMiEmpresa.BeginEdit();
//Modifico el valor del campo CompanyName
rowMiEmpresa["CompanyName"] = nuevonombre;
//Finalizo la edicion llamando al método EndEdit()
rowMiEmpresa.EndEdit();
```

- 7) Creamos un objeto command que será utilizado por el dataadapter para poder realizar los cambios necesarios en la base de datos. El objeto command que crearemos se llamará updcommand y le debemos indicar la cadena TSQL que utilizará para llamar a la BD y los parámetros que utiliza, tanto como su nombre, el tipo de dato, la longitud y el nombre del campo de nuestro datatable.

```
//Ahora creo un objeto Command que utilizare para
//guardar los cambios en la base de datos
SqlCommand updcommand = new SqlCommand();
//Le indico la coneccion
updcommand.Connection = myconn;
//Le indico la cadena TSQL
updcommand.CommandText =
    "UPDATE Customers SET CompanyName = @CompanyName WHERE CustomerID = @CustomerID";
//Indico los parametros que estoy utilizando.
//Como así tambien, el tipo de dato, la longitud del dato
//el nombre del campo del datatable
updcommand.Parameters.Add("@CompanyName", SqlDbType.NVarChar, 50, "CompanyName");
updcommand.Parameters.Add("@CustomerID", SqlDbType.NVarChar, 5, "CustomerID");
```

- 8) Por último adjuntamos éste objeto a nuestro DataAdapter (myada) para luego invocar su método .Update() e indicarle el datatable que tiene que actualizar contra la base de datos.

```
//Ahora adjunto el objeto updcommand al dataadapter
myadap.UpdateCommand = updcommand;
//Por último llamo al método .Update del DataAdapter
myadap.Update(dtEmpresas);
```

- 9) Hasta aquí hemos aprendido a consultar en la base de datos, traer los resultados y guardarlos en un objeto en memoria (datatable), modificar su contenido y volver a conectarnos a la base de datos para guardar los cambios.

- 10) De ésta manera se ve reflejado la importancia y trabajar en forma desconectada.

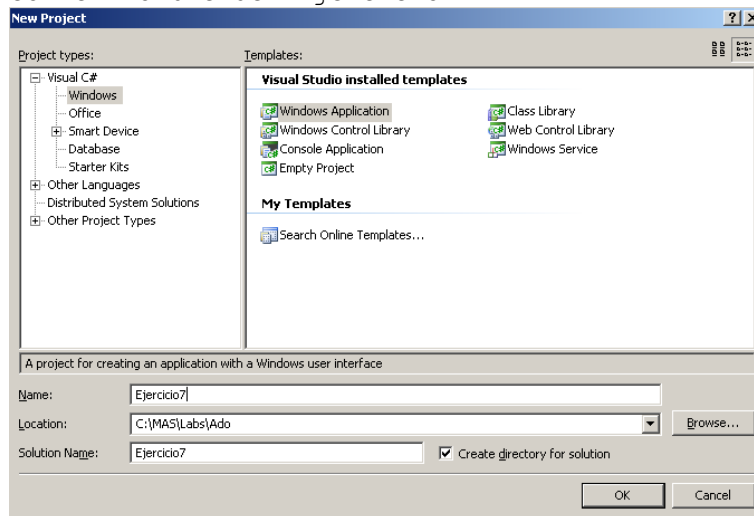
DataBinding

Duración Aproximada

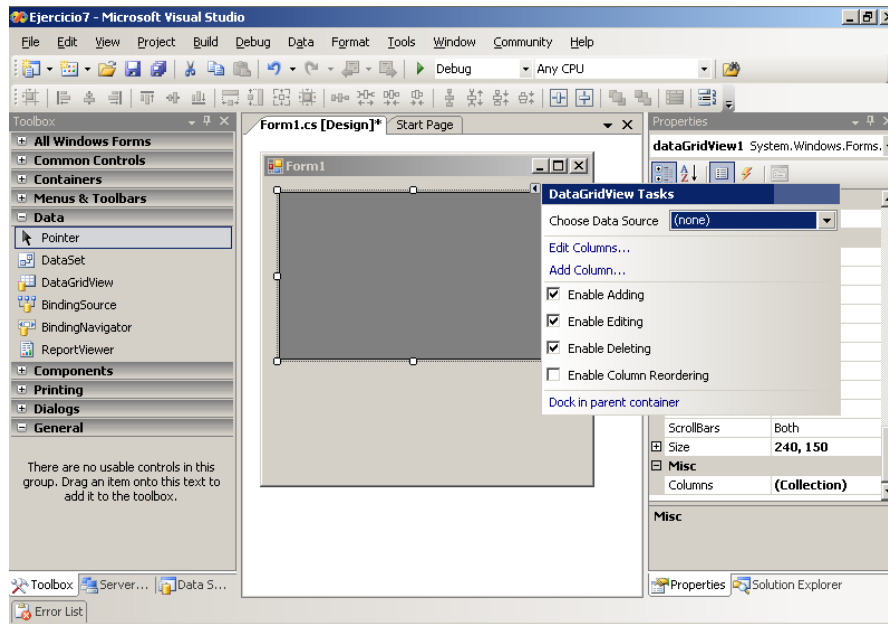
35 minutos

Pasos o Recomendaciones

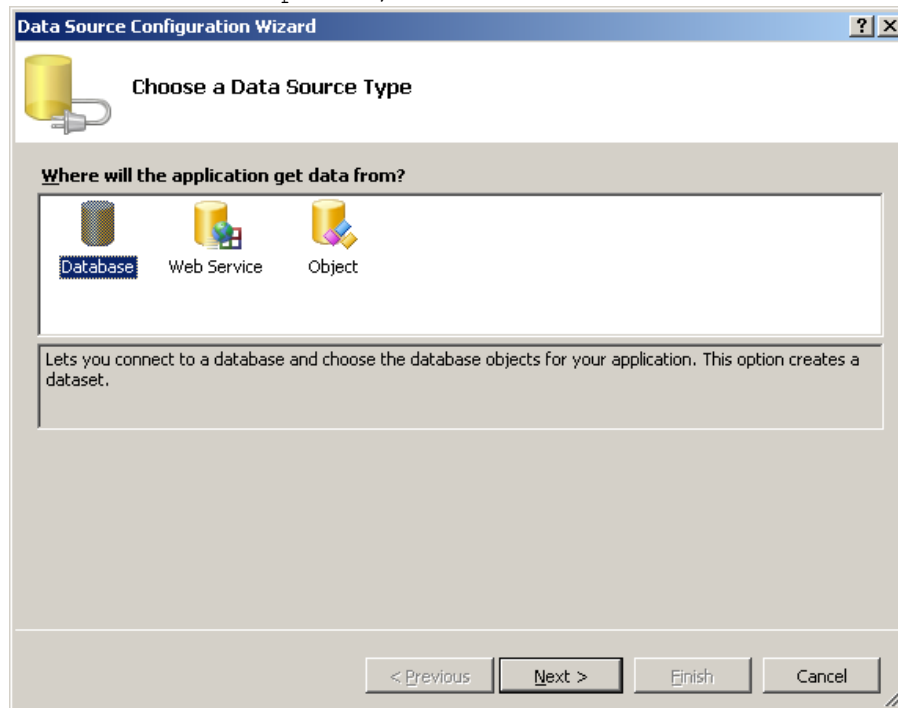
- 1) En éste ejercicio mostraremos básicamente como enlazar datos de un origen de datos con un control de WinForms,
- 2) Para ello crearemos un proyecto del tipo "Windows Application" con el nombre de "Ejercicio7"



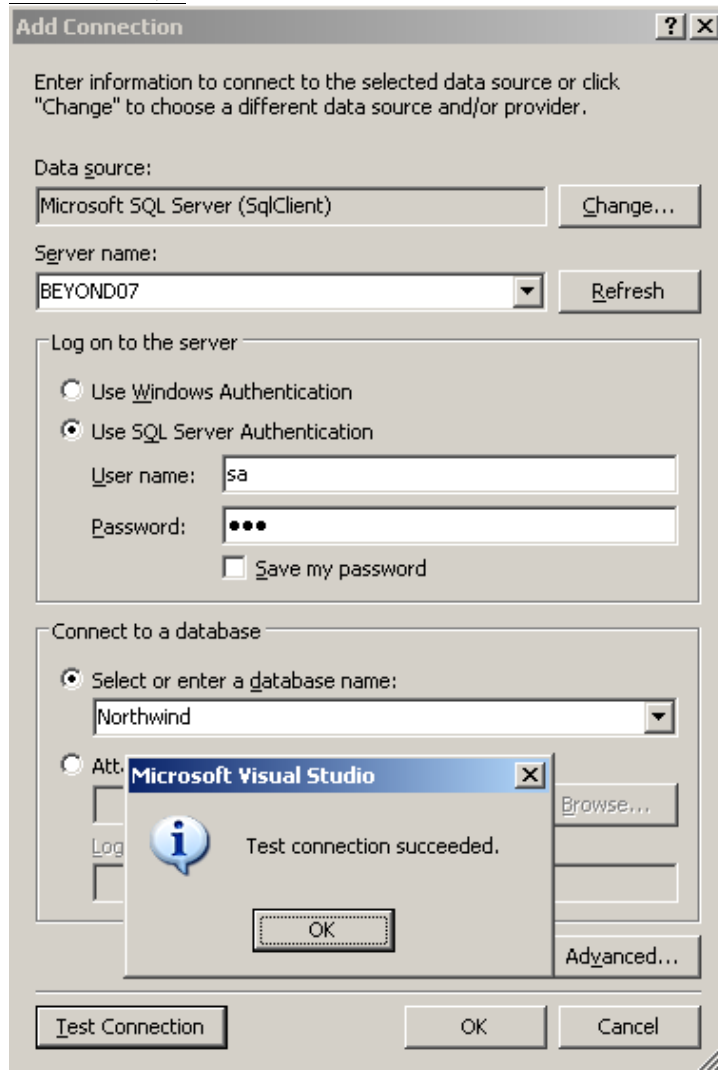
- 3) En primer lugar arrastrar desde nuestro ToolBox en la sección "Data", arrastraremos y colocaremos el control DataGridView sobre nuestro formulario "Form1", una vez colocado resaltará un cuadro de tareas correspondiente al DataGridView de donde seleccionaremos de "Choose Data Source" la opción "Add Project Data Source..."



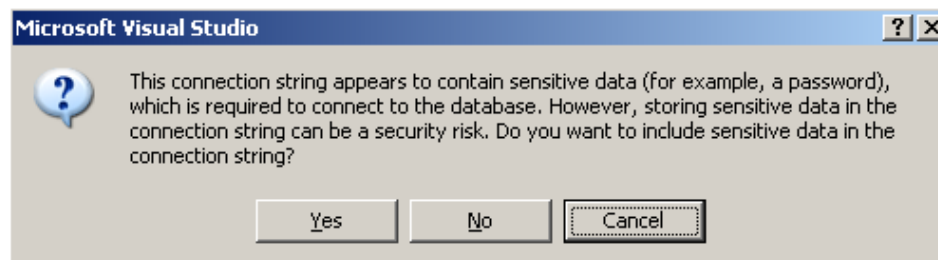
- 4) Una vez que seleccionamos "Add Project Data Source.." aparecerá un cuadro con templates, de donde seleccionaremos "DataBase"



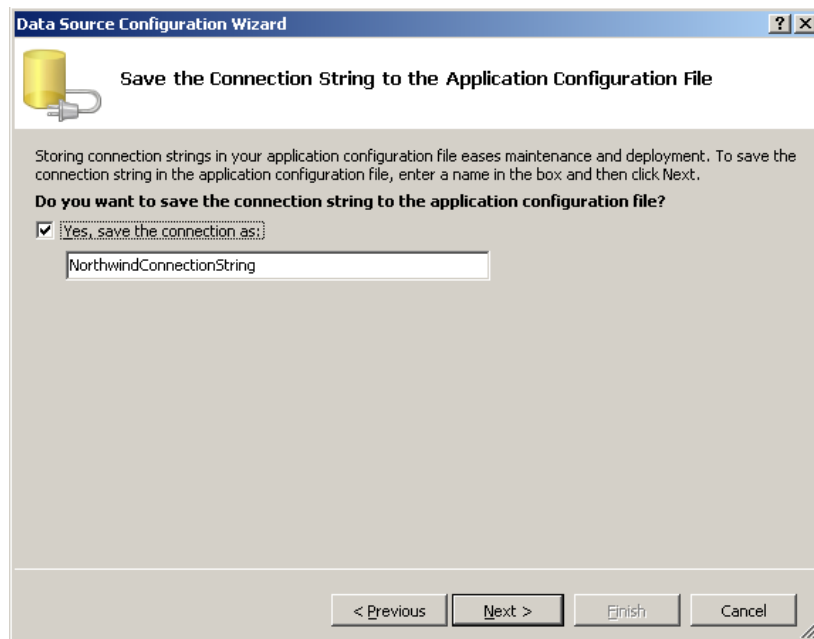
- 5) Seleccionamos DataBase, luego "Next", luego presionamos el boton "New Connection...". luego veremos un cuadro de dialogo para seleccionar el origen y la base de datos. (dependiendo del ambito donde se desarrolle el curso, se deberia verificar primeramente el nombre y la instancia de sql donde resida una base de datos Northwind).



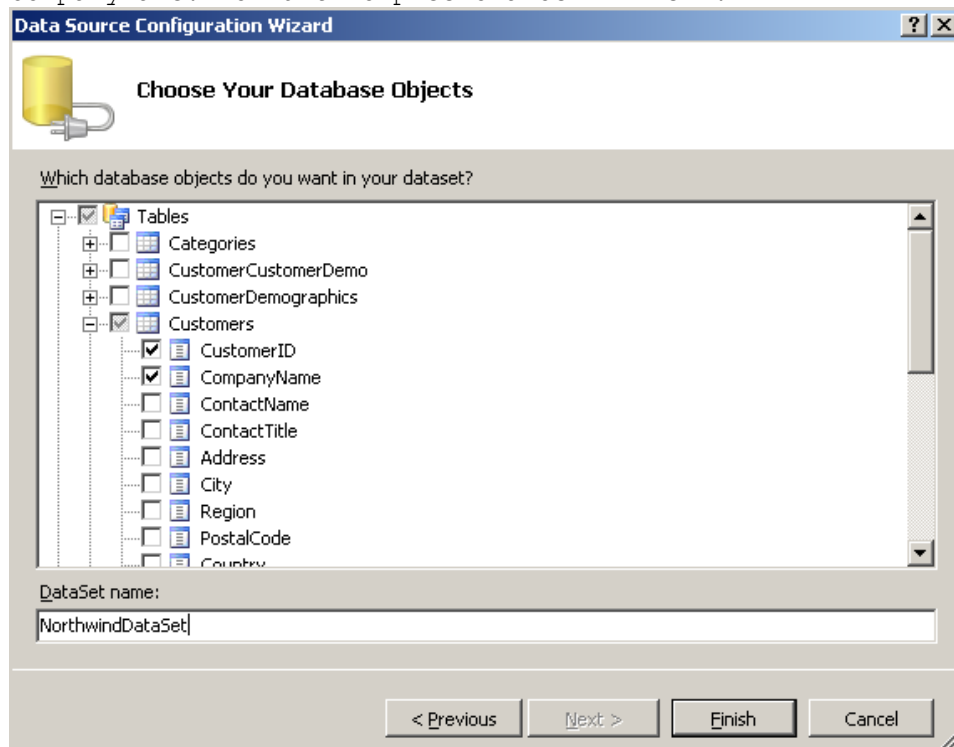
- 6) Una vez que hallamos presionado "Ok", VisualStudio nos notificará que es dentro de la cadena de coneccion hay datos importantes (nombre de usuario, contraseña), Presionamos "Yes" sobre el cuado de dialogo"



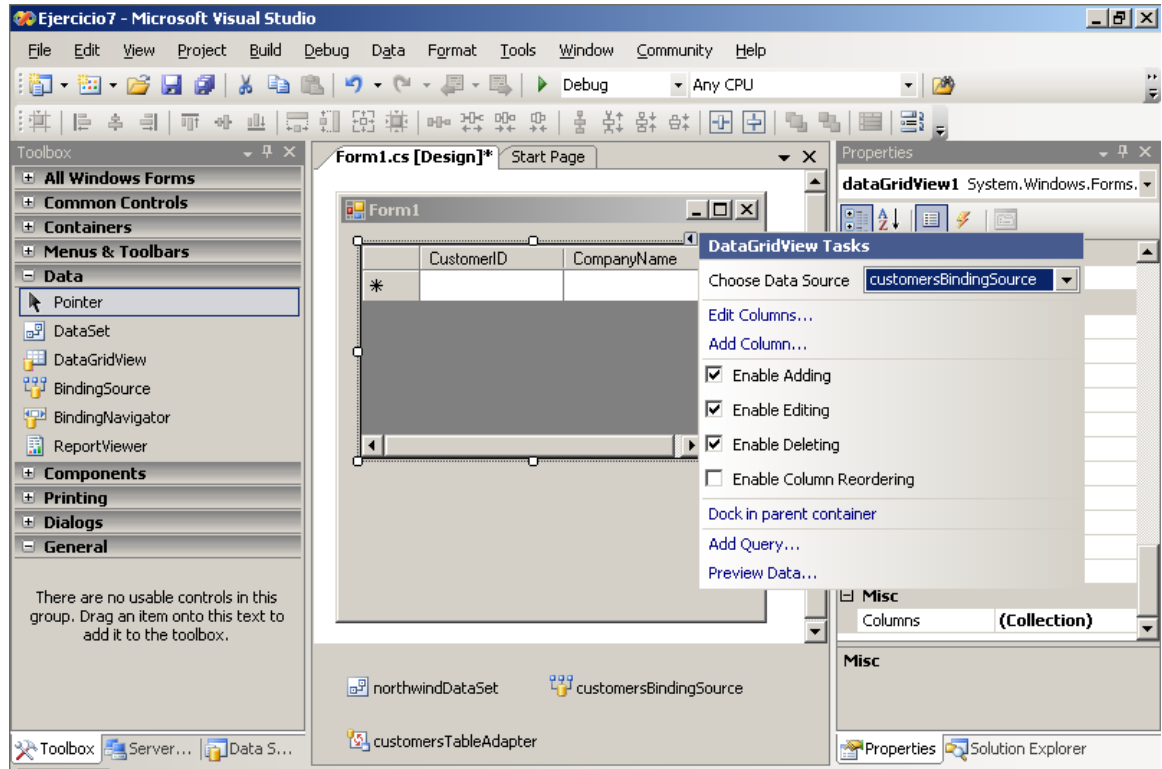
- 7) Ahora VisualStudio nos brinda la posibilidad de guardar el connection string dentro de un archivo externo de configuración, ya que si en el día de mañana cambia la ubicación del servidor, o las claves, etc, habria que entrar al código de la aplicación para modificar y volver a recompilar el programa. De ésta manera queda guardado en un archivo externo a la aplicación.



- 8) Por último seleccionaremos de Tablas, la tabla Customers, y de ahí seleccionaremos únicamente los campos CustomerID y CompanyName. Por último presionamos "Finish".



- 9) Luego de finalizado el asistente veremos que el control DataGridView esta enlazado con un control llamado "customerBindingSource", que a su vez este control esta enlazado con un northwindDataSet (un dataset tipado que ha creado) y un customerDataTable que utilizara para realizar las consultas y/o modificaciones sobre la base de datos.



- 10) Si presionamos F5 veremos que automáticamente se rellena el contenido en el control DataGridView
- 11) Fin del ejercicio.