

# ADO.NET

# ADO.NET - DataSet y DataTable

- Tanto **DataSet** como **DataTable** son clases que integran un grupo mayor de componentes denominado **ADO.NET**
- **ADO.NET** es una tecnología que está basada en **ActiveX® Data Objects** (ADO). Sin embargo no es no es una revisión de ADO sino una forma completamente nueva para manipular datos

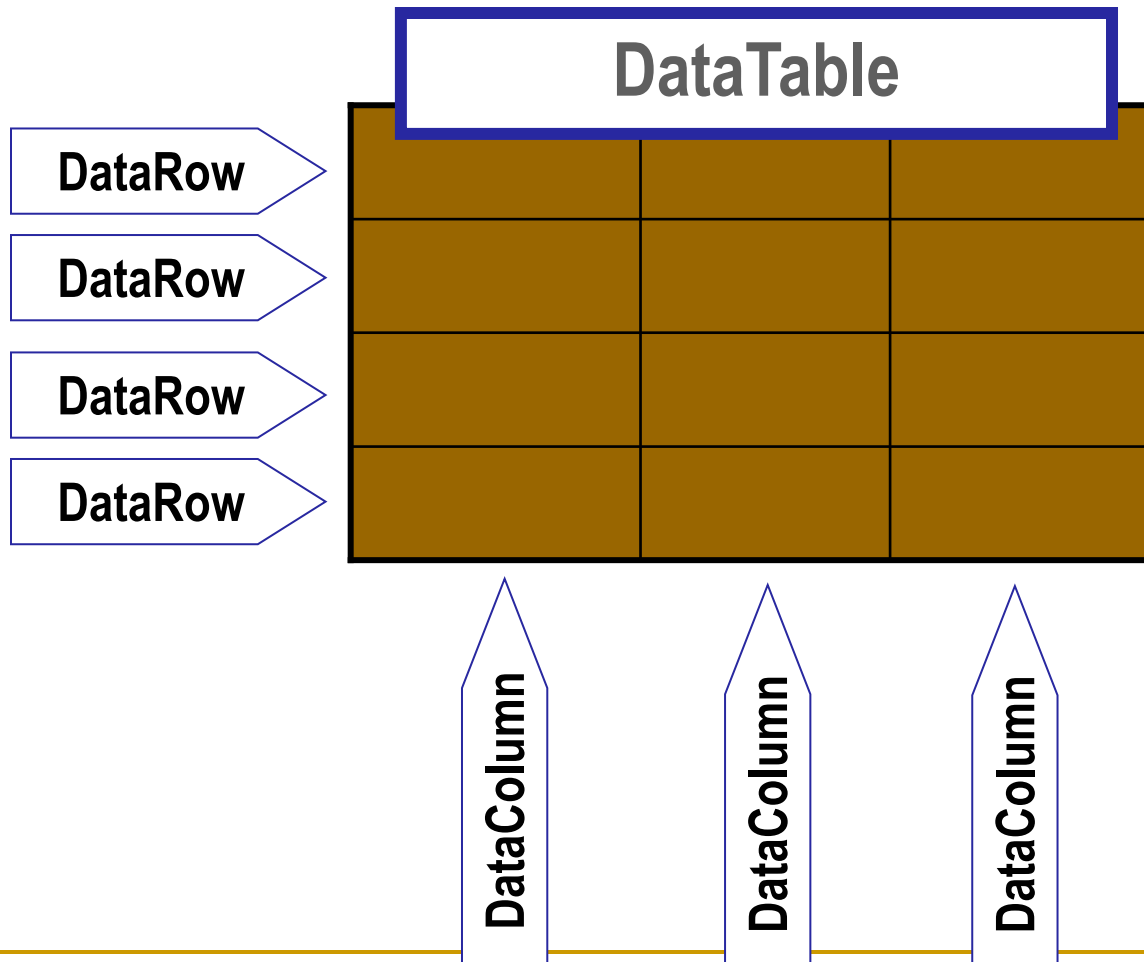
---

# ADO.NET

- Microsoft® ADO.NET es la tecnología que se utiliza para conectar las aplicaciones con **orígenes de datos**, como bases de datos **Microsoft SQL Server™** y archivos **XML**.
- **ADO.NET** está especialmente diseñado para trabajo con **entornos desconectados**, como internet, provee una forma flexible y simple para el desarrollo de aplicaciones que integran el acceso y la manipulación de datos.

# DataTable

## Tabla de datos en memoria



# DataTable

- Un objeto **DataTable** especifica una tabla de datos en memoria
- El **esquema de datos** de la tabla está definido mediante objetos  **DataColumn** y **Constraint**
- Los datos de la tabla se almacenan en una colección de objetos **DataRow**
- Tanto el esquema como los datos pueden ser creados **programáticamente**, recuperados desde una **base de datos** o cargados desde un **documento XML**
- Desconoce su origen de datos, por lo que funciona como una entidad independiente

---

# Creación de un DataTable

- Para crear un **DataTable** utilizaremos el operador **new**.
- El constructor acepta opcionalmente un argumento que permite establecer la propiedad **TableName** en el momento de la creación de la instancia.

# Creación de un DataTable

```
9      using System;
10     using System.Collections.Generic;
11     using System.Drawing;
12     using System.Windows.Forms;
13     using System.Data;
14
15
16     namespace PruebaADO
17     {
18         /// <summary>
19         /// Description of MainForm.
20         /// </summary>
21         public partial class MainForm : Form
22         {
23             DataTable dt=new DataTable("MiTabla");
24
25             public MainForm()
26             {
```

Agregar este espacio de nombre

Se está instanciando la tabla dt cuyo TableName es "MiTabla"

# Trabajando con Columnas

- La colección de todas las columnas de una tabla es accedida por medio de la propiedad **Columns** de la misma.
- Para agregar una columna a la tabla utilice el método **Add()** de la colección de columnas



# Trabajando con Columnas

Se agrega la primera columna de tipo entero

```
21 public partial class MainForm : Form
22 {
23     DataTable dt=new DataTable("MiTabla");
24
25     public MainForm()
26     {
27         // agregando una columna existente
28         DataColumn col = new DataColumn("Columna1", typeof(int));
29         dt.Columns.Add(col);
30
31         // agregando y creando una columna al mismo tiempo
32         dt.Columns.Add("Columna2", typeof(string));
33     }
34 }
```

Se agrega la segunda columna de tipo string

---

# Trabajando con Filas

- Un objeto **DataRow** representa una fila de una **DataTable**. Es el lugar donde realmente se almacenan los datos. La colección entera de filas de una tabla es accedida por medio de la propiedad **Rows** de la misma.
- Para agregar una fila a la tabla utilice el método **Add()** de la colección de filas.

# Trabajando con Filas

```
public MainForm()  
{  
    // agregando una columna existente  
    DataColumn col = new DataColumn("Columna1", typeof(int));  
    dt.Columns.Add(col);  
  
    // agregando y creando una columna al mismo tiempo  
    dt.Columns.Add("Columna2", typeof(string));  
  
    // agregando una fila existente  
    DataRow fila=dt.NewRow();  
    fila["Columna1"]=10;  
    fila["Columna2"]="diez";  
    dt.Rows.Add(fila);  
  
    //agregando y creando una fila al mismo tiempo  
    dt.Rows.Add(20, "veinte");  
}
```

Devuelve una fila nueva que se corresponde con el esquema de la tabla

Con la lista de parámetros (que deben coincidir con el esquema de la tabla) se crea una fila y se agrega

# Trabajando con Filas

- Otros métodos importantes que pueden aplicarse a la colección de filas son: **InsertAt()**, **Remove()**, **RemoveAt()** y **Clear()**

```
// inserta una nueva fila como primer elemento de la colección  
dt.Rows.InsertAt(fila, 0);
```

```
// remueve una fila de la colección  
dt.Rows.Remove(fila);
```

```
// remueve la primera fila de la colección  
dt.Rows.RemoveAt(0);
```

```
// remueve todas las filas de la tabla  
dt.Rows.Clear();
```

# Trabajando con Filas

- Agregue un **TextBox** al formulario. Establezca su propiedad **MultiLine** en **true**
- Codifique lo siguiente en el constructor luego de la invocación **InitializeComponent()**

```
InitializeComponent();
```

```
foreach (DataRow f in dt.Rows)
{
    textBox1.Text += f["Columna2"] + "\r\n";
}
```

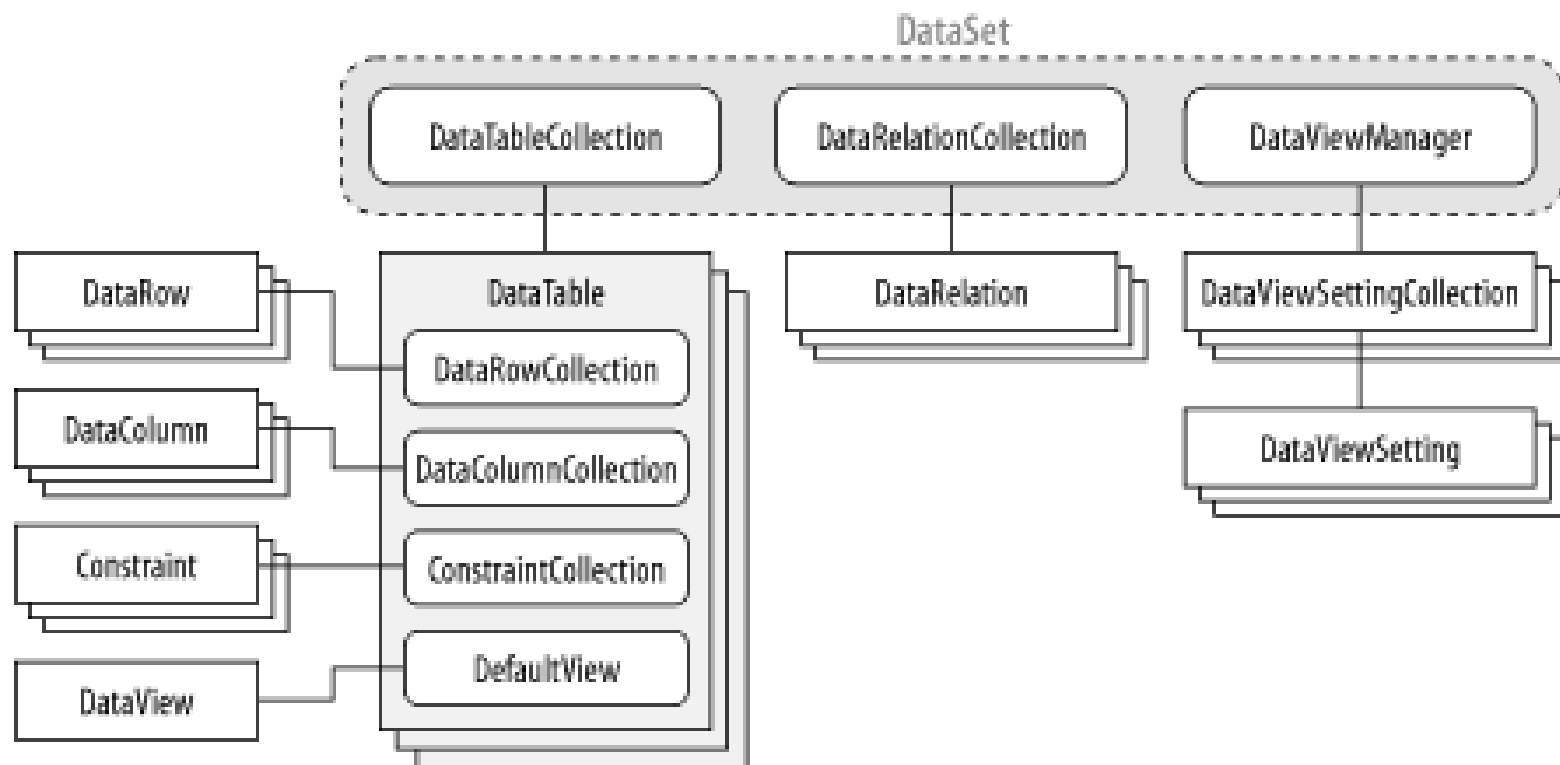
Ejecute y compruebe  
su funcionamiento

# DataSet

- Un **DataSet** almacena información en un **entorno desconectado**. Podemos imaginarlo como una base de datos en memoria.
- El objeto **DataSet** almacena los datos en uno o más **DataTables**.
- Cada **DataTable** puede ser llenado con datos de un sólo origen de datos. También puede establecer relaciones entre dos **DataTables** mediante el uso de un objeto **DataRelation**.

# DataSet

## The DataSet class



# DataSet

- La forma más simple de crear un **DataSet** es utilizar el operador **new**.
- El constructor acepta opcionalmente un argumento que permite establecer la propiedad **DataSetName** en el momento de la creación de la instancia.
- Un **DataSet** puede ser creado a partir de otro **Dataset** por medio de los métodos **Copy()** o **Clone()**



# DataSet

```
namespace PruebaADO
{
```

```
    /// <summary>
    /// Description of MainForm.
    /// </summary>
```

```
    public partial class MainForm : Form
    {
```

```
        DataSet ds= new DataSet("MiDataSet");
```

```
    public MainForm()
    {
```

```
        DataTable dt=new DataTable("MiTabla");
```

```
        // agregando una columna existente
```

```
        DataColumn col = new DataColumn("Columna1", typeof(int));
        dt.Columns.Add(col);
```

Se está instanciando el DataSet ds cuyo DataSetName es "MiDataSet"

Se ha movido la declaración de la tabla dt para que sólo sea accesible desde el constructor Mainform()

---

# Trabajando con DataSet

- Un **DataSet** posee una colección de tablas accedida por la propiedad **Tables**.
- Las tablas son agregadas al **DataSet** utilizando el método **Add()** de la colección de tablas.
- También puede agregar un arreglo de tablas utilizando el método **AddRange()**.

# Trabajando con DataSet

```
34
35      // agregando una fila existente
36      DataRow fila=dt.NewRow();
37      fila["Columna1"]=10;
38      fila["Columna2"]="diez";
39      dt.Rows.Add(fila);
40
41      //agregando y creando una fila al mismo tiempo
42      dt.Rows.Add(20,"veinte");
43
44      //agregando la tabla dt al dataset ds
45      ds.Tables.Add(dt);
46      //
47      // The InitializeComponent() call is required for Windows Forms
48      //
49      InitializeComponent();
```

# Visualizando una tabla en un formulario



# Visualizando una tabla en un formulario

```
//agregando y creando una fila al mismo tiempo
dt.Rows.Add(20, "veinte");

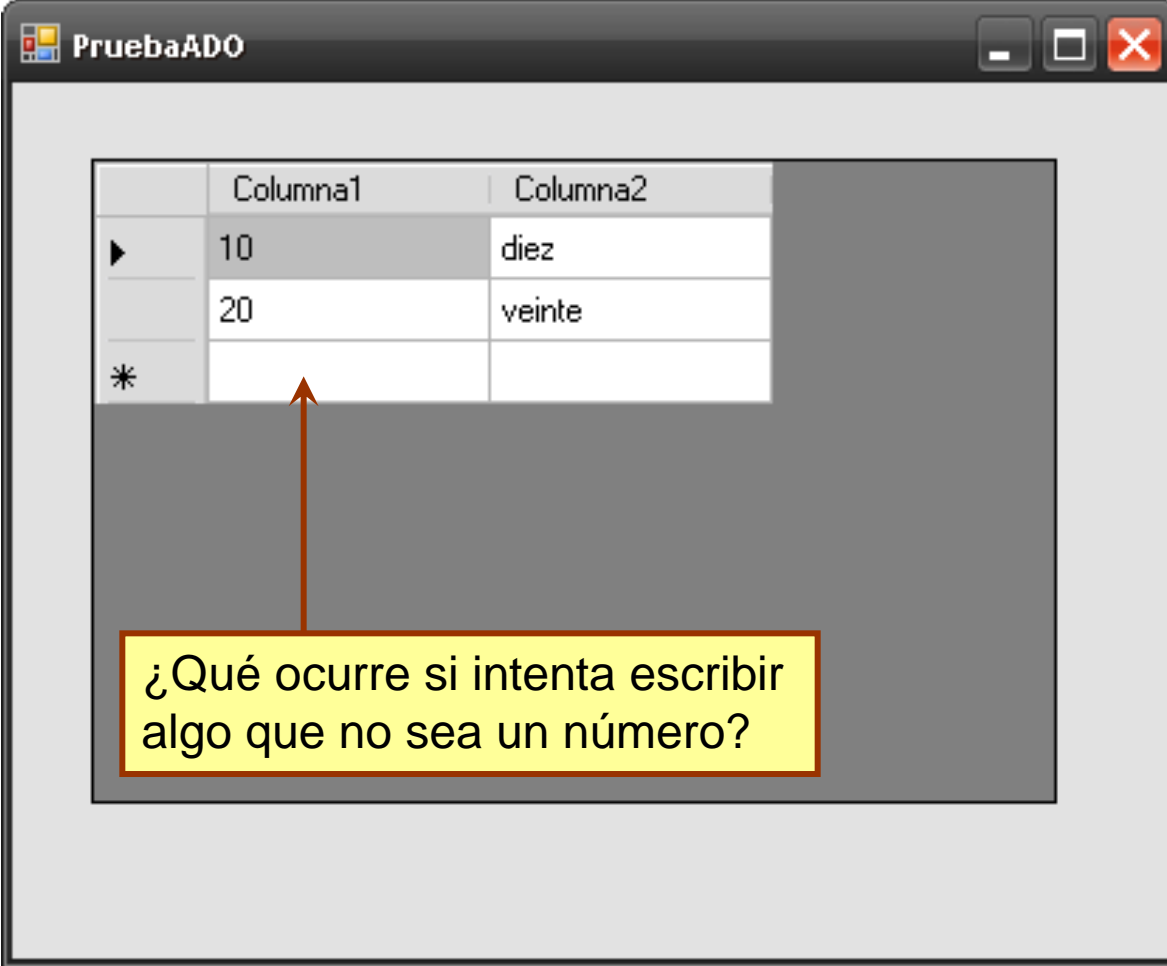
//agregando la tabla dt al dataset
ds.Tables.Add(dt);
//
// The InitializeComponent() call is required for Windows Forms des:
//
InitializeComponent();

//
// TODO: Add constructor code after the InitializeComponent() call.
//
}
```

Enlace la grilla a la tabla en el evento FormLoad

```
void MainFormLoad(object sender, EventArgs e)
{
    dataGridView1.DataSource=ds.Tables["MiTabla"];
}
```

# Visualizando una tabla en un formulario



PruebaADO

	Columna1	Columna2
▶	10	diez
	20	veinte
*		

¿Qué ocurre si intenta escribir algo que no sea un número?

# Ejercicio de programación

- Cree y rellene programáticamente las tablas ALUMNOS y NOTAS para visualizarlas en un formulario como se aprecia en la imagen

Tipos `string` e `int`

TABLA ALUMNOS

	Nombre	nro
▶	Perez,Rodriguez	1024
	Garía Néstor	877
	Rodriguez, Juan	912
*		

Tipos `int` y `float`

TABLA NOTAS

	nro	nota
▶	1024	6,5
	1024	6,7
	1024	8,5
*		

# Ejercicio de programación

```
DataTable dt=new DataTable("Alumnos");
dt.Columns.Add("Nombre",typeof(string));
dt.Columns.Add("nro",typeof(int));
dt.Rows.Add("Perez,Rodriguez",1024);
dt.Rows.Add("Garía Néstor",877);
dt.Rows.Add("Rodriguez, Juan",912);
ds.Tables.Add(dt);
dt=new DataTable("notas");
dt.Columns.Add("nro",typeof(int));
dt.Columns.Add("nota",typeof(double));
dt.Rows.Add(1024,6.5);
dt.Rows.Add(1024,6.7);
dt.Rows.Add(1024,8.5);
dt.Rows.Add(877,9);
dt.Rows.Add(877,10);
dt.Rows.Add(877,5);
dt.Rows.Add(912,9);
dt.Rows.Add(912,5);
dt.Rows.Add(912,7.5);
ds.Tables.Add(dt);
```



# Ejercicio de programación

- Se pretende establecer una relación Maestro / Detalle.
- Agregue un componente **BindingSource** al formulario. Estos componentes se utilizan como intermediarios entre un conjunto de datos y los controles del formulario.
- Agregue el siguiente código para conectar el control DataGridView

```
bindingSource1.DataSource = ds.Tables["Alumnos"];  
dataGridView1.DataSource = bindingSource1;
```

# Ejercicio de programación

- Agregue programáticamente un objeto **DataRelation** al **DataSet** para relacionar las columnas **nro** de ambas tablas

```
DataColumn colPadre=ds.Tables["Alumnos"].Columns["nro"];  
DataColumn colHija=ds.Tables["Notas"].Columns["nro"];  
DataRelation relacion=new DataRelation("relacion",colPadre,colHija);  
ds.Relations.Add(relacion);
```

# Ejercicio de programación

- Finalmente enlace la segunda grilla de la siguiente manera:

```
dataGridView2.DataSource=bindingSource1;  
dataGridView2.DataMember="relacion";
```