Controles

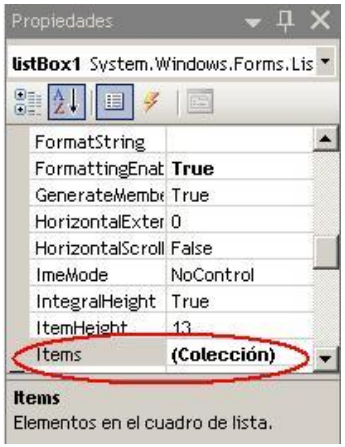
ListBox

ComboBox

CheckedListBox

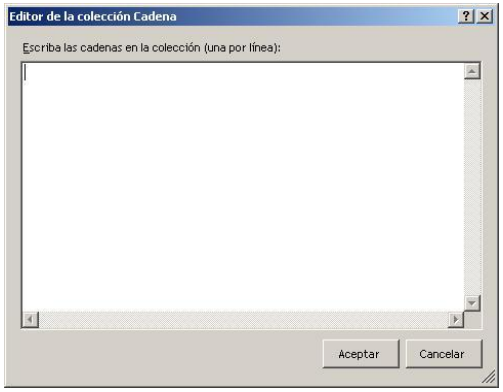
Introducción

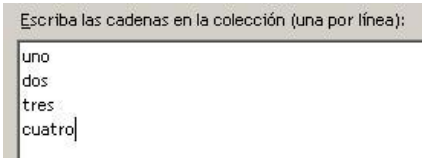
- Todos los entornos de desarrollo visuales disponen de una serie de controles que basan su gestión y funcionamiento en el empleo de **listas** (*colecciones de cadenas*).
- Algunos de ellos, como es el caso de ***listBox*** y ***comboBox*** disponen de una funcionalidad y uso generalizado en las aplicaciones visuales de escritorio, web, para dispositivos móviles, etc…
- Para introducirnos es el manejo de colecciones de cadenas, emplearemos el control básico **listBox** y la clase **listBox.ObjectCollection**.
- Unos (*la mayoría*) lo hacen empleando la propiedad **Items**, otros, manejan las colecciones de cadenas empleando otra propiedad, cuyo nombre, no desvirtúa lo esencial de su gestión y manejo



Introducción

- Como podemos observar, la propiedad Items se nos nuestra con la entrada: **(Colección)**, indicándonos su posible contenido. Si seleccionamos esta opción, a la derecha de esta expresión, aparecerá un botón que en su interior contiene tres puntos suspensivos que al ser pulsado dará entrada a la ejecución de un Editor de colección de cadenas.
- Los botones que se presentan con este aspecto, basarán su funcionamiento en la invocación de la ejecución de algún tipo de recurso. En este caso, al pulsar sobre el botón, aparecerá un **Editor de cadenas**.





Como podemos observar en las imágenes, cuando finalizamos el ciclo de ejecución, los **Items** (*colección de cadenas*) se habrán añadido al control **listBox**.

Introducción

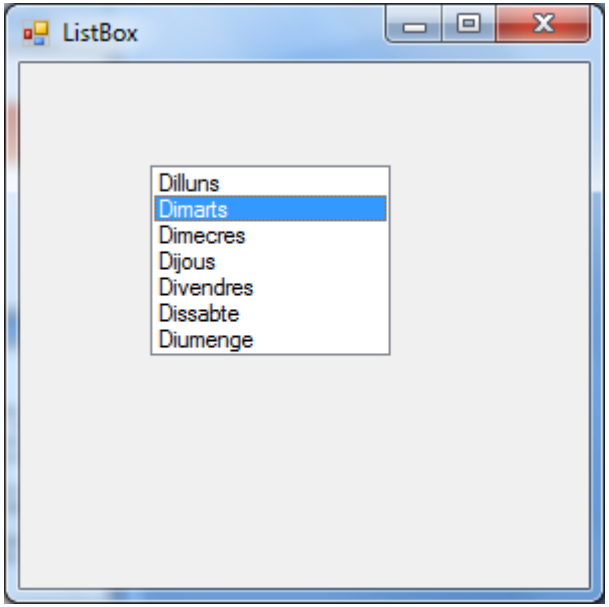
- Claro está que todo este proceso lo hemos realizado '*en tiempo de diseño*', y así es que todo esto no transmite mucha sensación de su potencial técnico y de gestión.
- Evidentemente, sólo emplearemos la asignación de colecciones de cadenas '*en tiempo de diseño*':
  - Cuando su contenido no vaya a cambiar, ó
  - Si queremos partir de una serie de valores iniciales.
- En cualquier otro caso, debemos aprender a asignar y manejar estos **Items** '*en tiempo de ejecución*'.

ListBox (Clase)

- Una lista es un control en el que se pueden mostrar varios registros o líneas, teniendo uno o varios de ellos seleccionado(s). Si en la lista hay más registros de los que se pueden mostrar al mismo tiempo, se añade automáticamente una **scrollBar**.
- **La propiedad Items** es una colección de objects que permite definir el contenido de la lista en modo de diseño a través de la ventana de propiedades.
- **Items** permite también acceder a los elementos de la lista en tiempo de ejecución, para utilizar y/o cambiar su valor:
  - **Listbox.Items[i] accede al contenido del elemento i del listbox**

ListBox (Clase)

- **Añadir** elementos en un listBox. En el ejemplo se carga en el evento private void Form1_Load(object sender, EventArgs e).

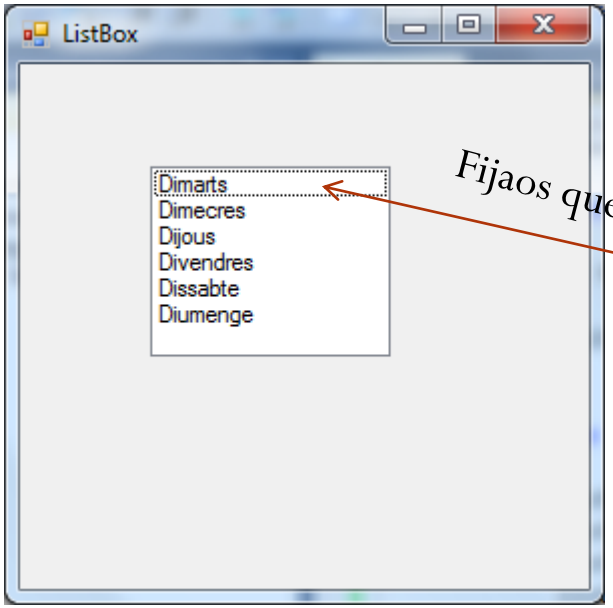



```
private void Form1_Load(object sender, EventArgs e)
{
    listBox1.Items.Add("Dilluns");
    listBox1.Items.Add("Dimarts");
    listBox1.Items.Add("Dimecres");
    listBox1.Items.Add("Dijous");
    listBox1.Items.Add("Divendres");
    listBox1.Items.Add("Dissabte");
    listBox1.Items.Add("Diumenge");
}
```

Fijaros que es una cadena, por lo tanto, podriamos cargarlo por medio de un textbox , una clase,etc…

ListBox (Clase)

- Para **eliminar** registros…



Fijaos que no esta el "Dilluns"

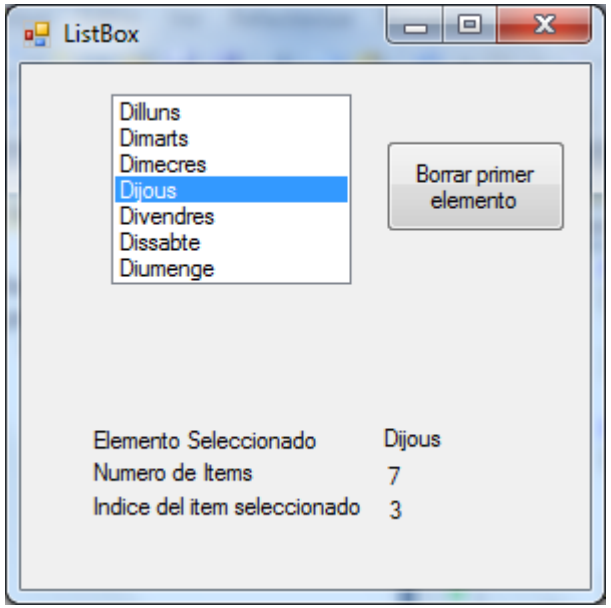
```
private void Form1_Load(object sender, EventArgs e)
{
    listBox1.Items.Add("Dilluns");
    listBox1.Items.Add("Dimarts");
    listBox1.Items.Add("Dimecres");
    listBox1.Items.Add("Dijous");
    listBox1.Items.Add("Divendres");
    listBox1.Items.Add("Dissabte");
    listBox1.Items.Add("Diumenge");

    listBox1.Items.RemoveAt(0);
}
```

Índice del elemento que quiero eliminar.
Podemos borrar de manera dinámica.

ListBox (Clase)

- Otras propiedades
  - Tamaño de la lista
    - **listBox1.Items.Count**
  - Elemento de la lista seleccionado actualmente
    - **listBox1.SelectedItem**
  - Índice del elemento seleccionado actualmente
    - **listBox1.SelectedIndex**

ListBox (Clase)



Si hacemos dobleclick sobre el listBox se nos crea el evento, así de manera automática se nos van actualizando los contenidos de las labels. Podriamos hacerlo desde cualquier otro evento, por ejemplo, al pulsar un botón.

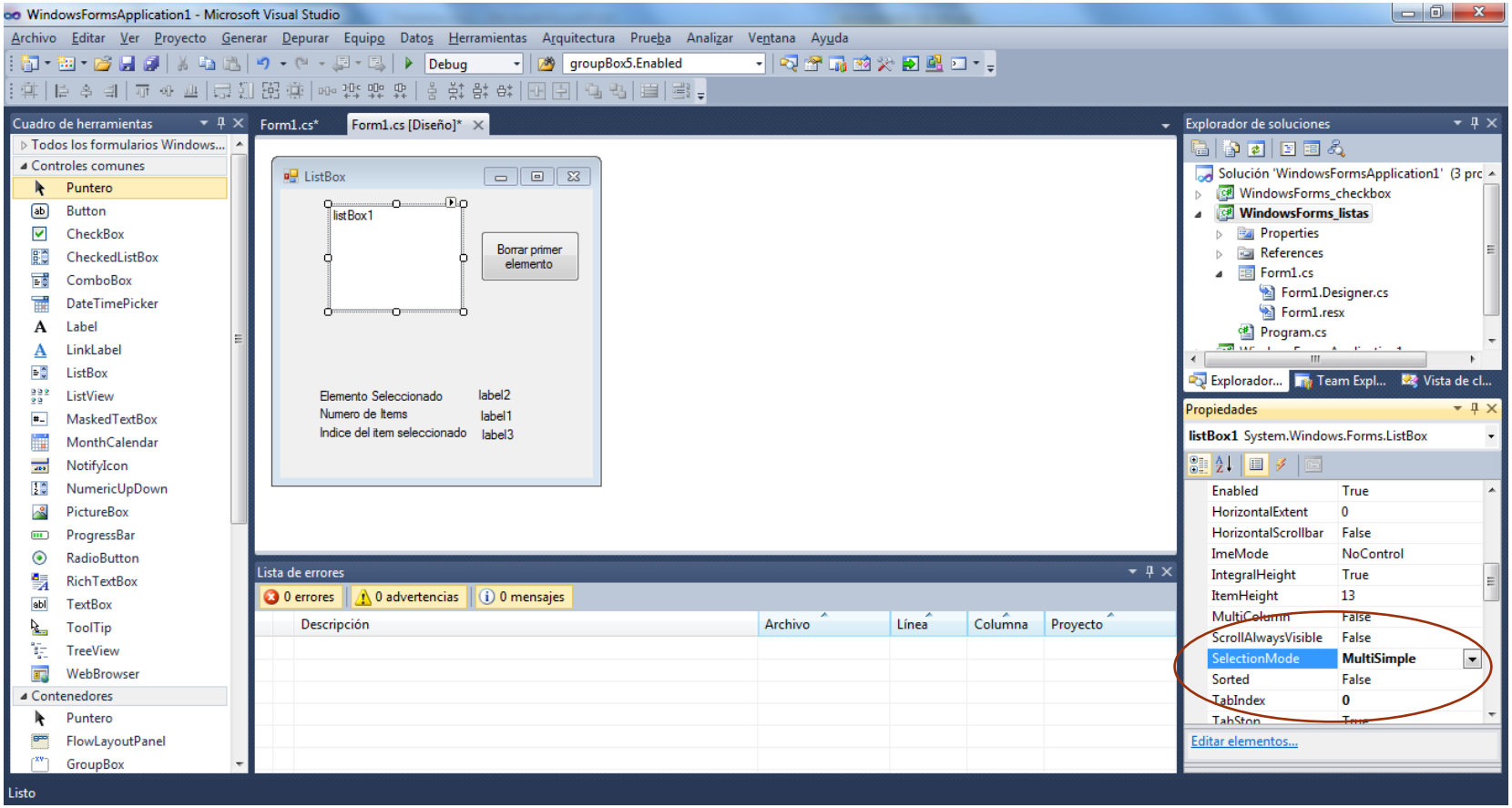
```
private void listBox1_SelectedIndexChanged(object sender, EventArgs e)
{
    label1.Text = listBox1.Items.Count.ToString();
    label2.Text = listBox1.SelectedItem.ToString();
    label3.Text = listBox1.SelectedIndex.ToString();
}
```

¿Qué diferencia hay?

```
private void listBox1_Click(object sender, EventArgs e)
{
    label1.Text = listBox1.Items.Count.ToString();
    label2.Text = listBox1.SelectedItem.ToString();
    label3.Text = listBox1.SelectedIndex.ToString();
}
```

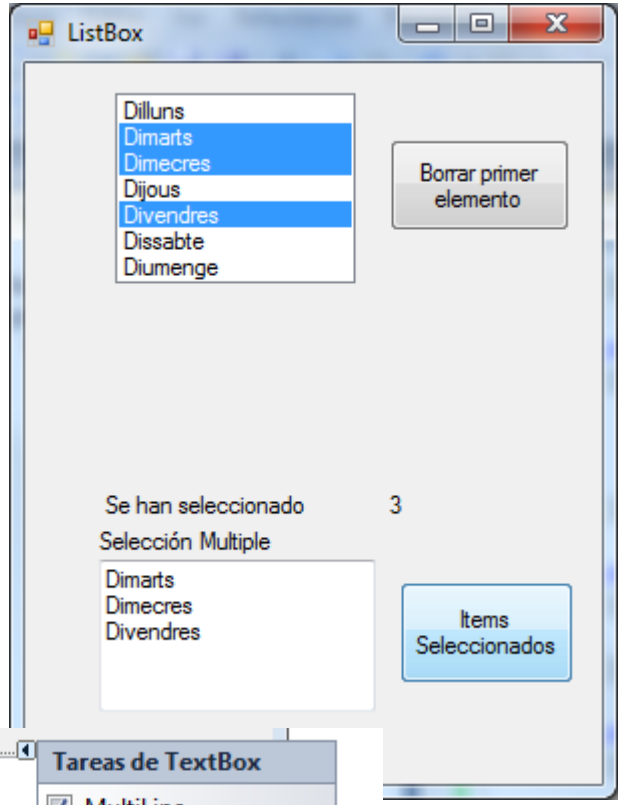
ListBox (Clase)

- Las listas permiten seleccionar más de un elemento poniendo en la propiedad ***SelectionMode el*** valor adecuado.



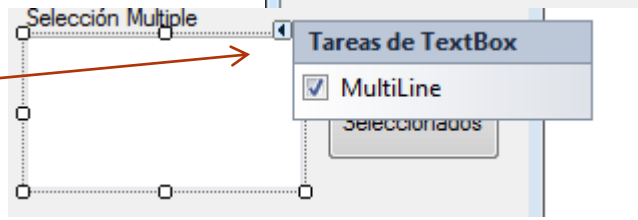
ListBox (Clase)

- Cantidad de elementos seleccionados en caso de selección múltiple
  - **listBox1.SelectedItems.Count**
- Devuelve true o false dependiendo si esta o no seleccionado
  - **listBox1.GetSelected(i)**
- Para acceder a cada uno de ellos lo haríamos como en la selección individual utilizando los métodos anteriores.




```
for (int i = 0; i < listBox1.Items.Count; i++)
        {
           if(listBox1.GetSelected(i))
              textBox1.Text += istBox1.Items[i]+"\r\n";
        }
```

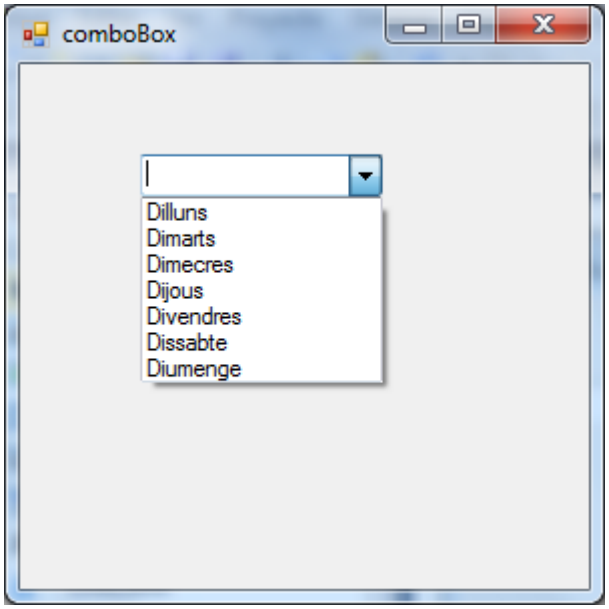
Para conseguir este efecto en un textbox
Pinchar en la flecha y seleccionar Multiline,
Lo redimensionais y poneis el retorno de carro y salto de linea tb se hubiese podido utilizar el
textBox1.AppendText(ListBox1.Items[i]+"\r\n");



ComboBox Class

- Un ComboBox tiene muchas cosas en común con una lista. Tiene una propiedad Items que es igual a los de la lista. Cambia únicamente la apariencia del control.
- Así tendremos…
- Para añadir

```
private void Form1_Load(object sender, EventArgs e)
{
    comboBox1.Items.Add("Dilluns");
    comboBox1.Items.Add("Dimarts");
    comboBox1.Items.Add("Dimecres");
    comboBox1.Items.Add("Dijous");
    comboBox1.Items.Add("Divendres");
    comboBox1.Items.Add("Dissabte");
    comboBox1.Items.Add("Diumenge");
}
```



- Para seleccionar

```
private void comboBox1_SelectedIndexChanged(object sender, EventArgs e)
{
    label1.Text = comboBox1.SelectedItem.ToString();
}
```

- Para borrar

```
comboBox1.Items.Remove(comboBox1.SelectedItem);
comboBox1.Items.RemoveAt(0);
comboBox1.Items.Remove("Dilluns");
```

El seleccionado

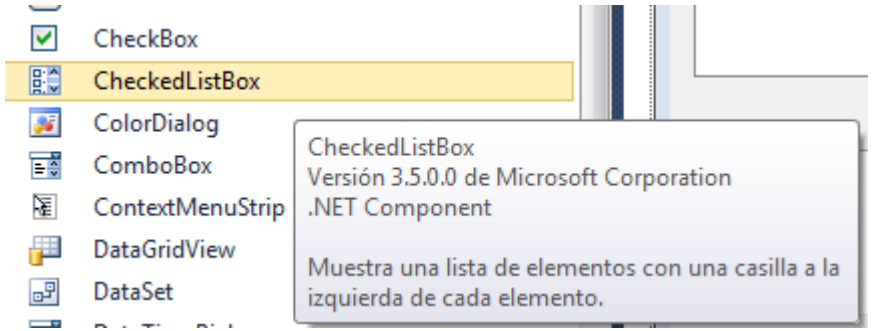
El primer elemento

Uno en concreto

CheckedListbox

- Al igual que los anteriores tendremos:
  - Añadir

```
private void Form1_Load(object sender, EventArgs e)
{
    checkedListBox1.Items.Add("Dilluns");
    checkedListBox1.Items.Add("Dimarts");
    checkedListBox1.Items.Add("Dimecres");
    checkedListBox1.Items.Add("Dijous");
    checkedListBox1.Items.Add("Divendres");
    checkedListBox1.Items.Add("Dissabte");
    checkedListBox1.Items.Add("Diumenge");
}
```



  - Borrar

```
checkedListBox1.Items.Remove(checkedListBox1.CheckedItems[i]);
```

  - Seleccionados

```
int i;
for (i = 0; i < checkedListBox1.CheckedItems.Count; i++)
{
    textBox1.AppendText(checkedListBox1.CheckedItems[i] + "\r\n");
}
```