

# ASP.NET Windows Forms

# ¿ Qué es Windows Forms ?

- ❖ Windows Forms es un subconjunto de la .NET Framework Class Library que permite el desarrollo de aplicaciones de escritorio ricas bajo Microsoft Windows.
- ❖ Incluye clases base, interfaces, enumeraciones y controles gráficos diversos.

# ¿Qué es un formulario ?

- ❖ Un formulario Windows Forms actúa como interfaz del usuario local de Windows.
- ❖ Los formularios pueden ser ventanas estándar, interfaces de múltiples documentos (MDI), cuadros de diálogo, etc.
- ❖ Los formularios son clases que exponen propiedades, métodos que definen su comportamiento y eventos que definen la interacción con el usuario.

# El diseñador de formularios

- Al momento de diseñar un formulario, el diseñador de Visual Studio Express escribe de forma automática el código que describe a cada uno de los controles y al propio formulario.
- El concepto de Partial class que incorpora .NET 2.0 permite separar el código de una clase en varios archivos fuentes diferentes.
- El diseñador de formularios utiliza esta técnica para escribir en un archivo aparte todo el código que él mismo genera.
- Esto permite organizar más claramente el código, manteniendo separada la lógica de la aplicación en un archivo diferente.

# Generalidades (1/2)

- El objeto Form es el principal componente de una aplicación Windows.
- Algunas de sus propiedades admiten valores de alguno de los tipos nativos de .NET
  - Ejemplo Código C#
    - miForm.ShowInTaskBar = false;
    - miForm.Opacity = 0.83;
  - Ejemplo Código Visual Basic
    - miForm.ShowInTaskBar = False
    - miForm.Opacity = 0.83

# Generalidades (2/2)

❖ Otras propiedades requieren la asignación de objetos

- Ejemplo en C#

- miForm.Size = new Size(100, 100);
- miForm.Location = new Location(0, 0);

- Ejemplo en Visual Basic

- miForm.Size = New Size(100, 100)
- miForm.Location = New Location(0, 0)

# Métodos

## Show()

- Visualiza el formulario. Puede especificarse su formulario **Owner**.
  - Si un formulario A es **owner** (dueño) de otro B, el formulario B siempre se visualizará sobre el A, sin importar si otro formulario está activo.

## ShowDialog()

- Visualiza el formulario como cuadro de diálogo **Modal**.
  - Un formulario visualizado de forma **modal** no permite que otro formulario perteneciente a la misma aplicación tome foco. Esta opción es utilizada para mostrar cuadros de diálogo y focalizar la atención del usuario.

# Eventos (1/2)

## Manejadores de eventos

- Por cada evento soportado por el Form (o por cualquier otro objeto) es posible definir varios métodos manejadores.
- A su vez, un método manejador puede controlar eventos disparados por diferentes objetos.

# Eventos (2/2)

## Ejemplos:

- Código C#

```
// Varios manejadores para un evento  
this.Click += new EventHandler(MetodoManejador1);  
this.Click += new EventHandler(MetodoManejador2);  
// Un mismo manejador para diferentes eventos  
this.Load += new EventHandler(ManejadorCentralizado);  
this.Activated +=new EventHandler(ManejadorCentralizado);
```

- Código Visual Basic

```
' Varios manejadores para un evento  
AddHandler Me.Click, AddressOf MetodoManejador1  
AddHandler Me.Click, AddressOf MetodoManejador2  
' Un mismo manejador para diferentes eventos  
AddHandler Me.Load, AddressOf ManejadorCentralizado  
AddHandler Me.Activated, AddressOf ManejadorCentralizado
```

# Ciclo de vida del formulario

- ❖ Muchos de los eventos a los que responde el *objeto Form* pertenecen al **ciclo de vida** del formulario
- ❖ Entre estos eventos se encuentran los siguientes, en orden de ocurrencia:
  - Load: El formulario está en memoria, pero invisible.
  - Paint: Se “pinta” el formulario y sus controles.
  - Activated: El formulario recibe foco.
  - FormClosing: Permite cancelar el cierre.
  - FormClosed: El formulario es invisible.
  - Disposed: El objeto está siendo destruido.

# Trabajando con el Mouse

- ❖ El mouse puede ser controlado escribiendo código para alguno de estos eventos:
  - **MouseClick**
  - **MouseEnter**
  - **MouseMove**
- ❖ A través de los argumentos que reciben los manejadores de estos eventos se puede obtener:
  - La posición del puntero
  - Qué botón fue presionado
  - Cantidad de “pasos” que fue girada la rueda

# Trabajando con el Teclado

- ▣ El manejador del evento KeyPress informa a través del argumento e.KeyChar el código de la tecla presionada.
- ▣ Es posible cancelar el comportamiento por defecto asignando “true” al argumento e.Handled.
- ▣ Los argumentos que reciben los manejadores de los eventos KeyDown y KeyUp informan del estado de las teclas Alt, Ctrl y Shift.
- ▣ El evento HelpRequested es disparado cuando se presiona la tecla F1.

# Foco de controles y orden de tabulación

- ❖ El objeto Form expone diferentes propiedades, métodos y eventos que permiten controlar la navegabilidad del formulario:
  - Propiedad *CanFocus*: Indica si el control puede tomar foco.
  - Propiedad *Focused*: Indica si el control tiene el foco actualmente.
  - Método *Focus()*: “Mueve” el foco al objeto deseado.
- ❖ Orden de tabulación (Propiedad *TabIndex*)
  - En forma visual, desde el diseñador de formularios, es posible configurar el orden en el que el foco se irá moviendo por los controles.

# Controles de Windows (1/3)

- ➊ Gran parte del éxito de una aplicación Windows consiste en elegir y manejar adecuadamente los controles que ofrece .NET.
- ➋ Entre los controles nativos se encuentran controles totalmente nuevos y versiones mejoradas de sus pares de .NET 1.1.
- ➌ Nuevos controles como el control BindingSource mejoran notablemente el enlace de datos provenientes de muy diferentes fuentes de datos.

# Controles de Windows (2/3)

## ☒ MaskedEdit

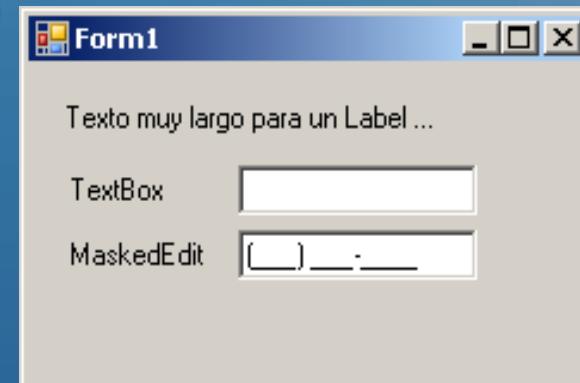
- Es un control que permite el uso de máscaras personalizadas para facilitar la entrada de datos.

## ☒ TextBox

- Cuadro de texto que, entre otras mejoras tiene la funcionalidad de auto completar.

## ☒ Label

- Si el texto ocupa más lugar que el largo del control, gracias a la nueva propiedad *AutoEllipsis* incorporada en .NET 2.0, el exedente se reemplaza automáticamente con tres puntos (...)



# Controles de Windows (3/3)

## DataGridView

- Es una versión mejorada del DataGrid control de .NET 1.1 con funcionalidad de modo “Virtual”. Permite enlazar datos originados en una Base de Datos a medida que se necesitan.

DataGridView

	Surname	Given Name	Date of Hire	Age	Gender	Marital Status	Salaried
▶	Goldberg	Jossef	February, 1998	56	M	M	<input checked="" type="checkbox"/>
	Duffy	Terri	March, 1998	44	F	S	<input checked="" type="checkbox"/>
	Higa	Sidney	March, 1998	59	M	M	<input type="checkbox"/>
	Maxwell	Taylor	March, 1998	59	M	M	<input type="checkbox"/>
	Ford	Jeffrey	March, 1998	59	M	S	<input type="checkbox"/>
	Brown	Jo	March, 1998	59	F	S	<input type="checkbox"/>
	Hartwig	Doris	April, 1998	59	F	M	<input type="checkbox"/>
	Campbell	John	April, 1998	59	M	M	<input type="checkbox"/>
	Grimp	Diane	April, 1998	59	F	M	<input type="checkbox"/>
	Johnson	Barry	February, 1998	59	M	S	<input type="checkbox"/>
	Krebs	Peter	January, 1999	32	M	M	<input checked="" type="checkbox"/>
	Erickson	Gail	February, 1998	63	F	M	<input checked="" type="checkbox"/>
	Ellerbrock	Ruth	February, 1998	59	F	M	<input type="checkbox"/>
	Dobney	JoLynn	January, 1998	59	F	S	<input type="checkbox"/>
	Mu	Zheng	January, 1999	31	M	S	<input type="checkbox"/>
	Bradley	David	January, 1998	40	M	S	<input checked="" type="checkbox"/>
	Komosinski	Paul	January, 1999	34	M	S	<input type="checkbox"/>

## TreeView

- Utilizando la nueva propiedad DrawMode es posible sobreescribir la manera en que el sistema operativo “dibuja” cada nodo del árbol.

# Controles Contenedores

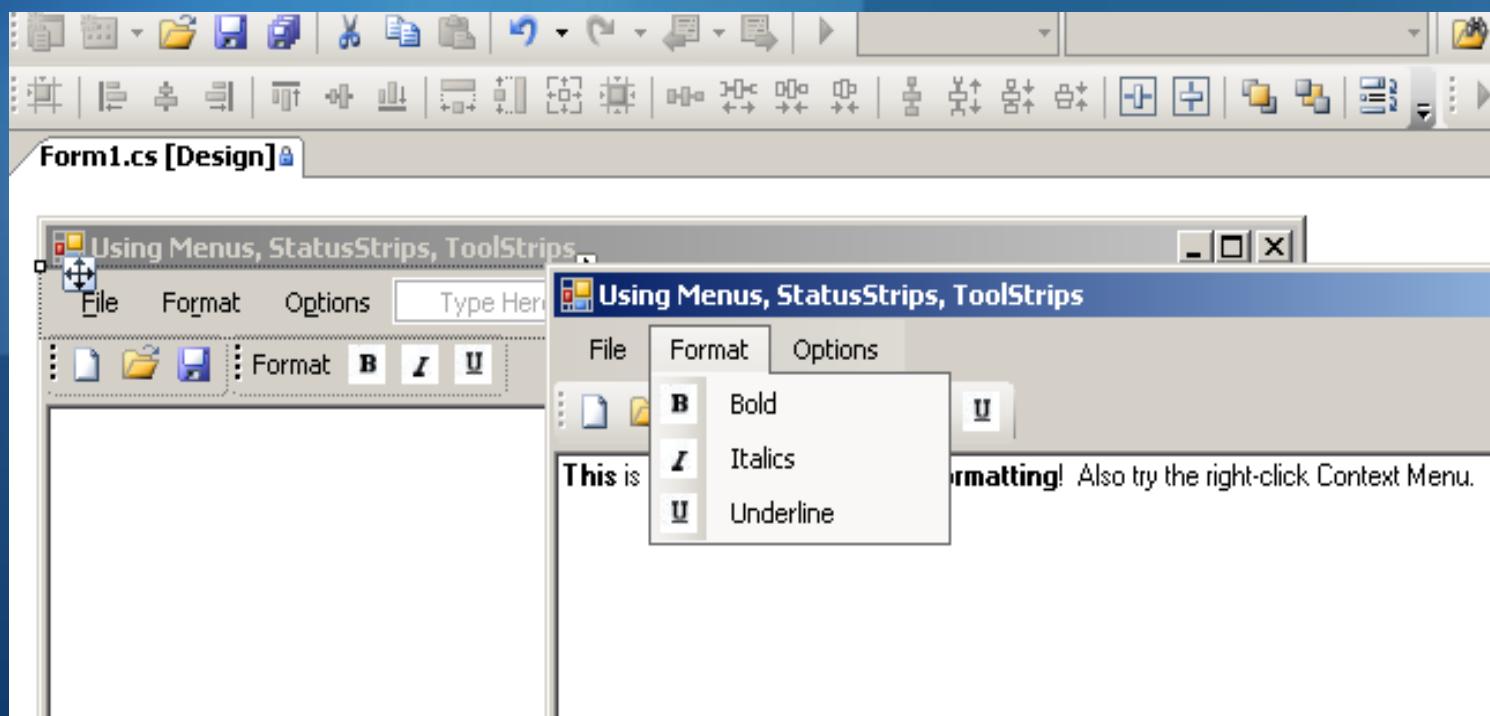
- ◉ Algunos controles como el propio Form, Panel o GroupBox heredan de la clase ContainerControl en lugar de hacerlo directamente de Control.
- ◉ Por este motivo, poseen una colección mediante la que se puede acceder a los controles que contiene.
- ◉ Sólo se puede acceder a los controles de nivel superior, no a todos los controles contenidos.

# Menú (1/2)

- ❖ El nuevo control **MenuStrip** provee un sistema de menú para un formulario.
- ❖ **MenuStrip** es contenedor de objetos como **ToolStripMenuItem**, **ToolStripComboBox**, **ToolStripSeparator**, **ToolStripTextBox**.
- ❖ El control **ContextMenuStrip** representa un menú que será mostrado al usuario cuando presione el botón derecho del mouse. También puede contener los mismos controles que **MenuStrip**.
- ❖ Las propiedades **MergeAction** y **MergeIndex** del objeto **ToolStripItem** permiten controlar la manera en que los menú de dos diferentes ventanas se “mezclarán”.

# Menú (2/2)

- En la imagen se ve una aplicación que utiliza los controles MenuStrip y ToolStrip. En segundo plano se ve el diseñador de formularios.



# Controles Extender Providers

- Son controles que, una vez colocados en un formulario, agregan nuevas propiedades a los otros controles existentes.
  - **ErrorProvider**: Permite asociar un error a un control mostrando un ícono que parpadea al lado de dicho control.
  - **HelpProvider**: Permite asociar a un control desde una simple cadena de texto un archivo Help que serán mostrados al presionar F1.
  - **ToolTip**: Es el clásico rectángulo que aparece asociado a un control y que es mostrado cuando el mouse se detiene sobre él.

# Herencia Visual

- ❖ Dado que un formulario Windows es como cualquier otra clase .NET, es posible aplicar herencia.
- ❖ Al heredar de un formulario base, además de sus miembros, se heredan todos los controles que en él se encuentren.
- ❖ Permite entre otras cosas:
  - Unificar el diseño de las interfaces de usuario.
  - Reutilizar funcionalidad de formularios similares.

# Diálogos Comunes

- ◉ Los cuadros de diálogo comunes permiten interacción con el usuario para ejecutar acciones comunes como abrir un archivo, configurar la impresión, seleccionar un color del sistema, etc.
- ◉ Sólo basta configurar algunas propiedades e invocar su método ShowDialog().
- ◉ Alguno de los controles que muestran estos diálogos son:
  - ColorDialog
  - PrintDialog
  - SaveDialog
  - OpenDialog

# Colecciones

- Enlace de un ComboBox a datos provenientes de un ArrayList:

- Código C#

```
System.Collections.ArrayList Paises =  
    new System.Collections.ArrayList();  
Paises.Add("Argentina");  
Paises.Add("Brasil");  
Paises.Add("Uruguay");  
comboBox1.DataSource = Paises;
```

- Código Visual Basic

```
Dim Paises As System.Collections.ArrayList = New __  
    System.Collections.ArrayList  
Paises.Add("Argentina")  
Paises.Add("Brasil")  
Paises.Add("Uruguay")  
comboBox1.DataSource = Paises
```

# Objeto BindingSource

- ❖ El objeto BindingSource permite el enlace de controles a datos provenientes de fuentes de datos (DataSource) de tres tipos
  - DataBase: Crea internamente un dataset.
  - WebService: Crea una referencia web a un servicio que es el que proporciona los datos
  - Object: Utiliza una clase de negocios como fuente de datos creando automáticamente una colección de elementos de esa clase.
- ❖ Usándolo junto a un control DataBindingNavigator y un DataGridView conforman un formulario de ABM sin escribir código alguno.