

Introducción a C# .NET

Módulo 5

Programación visual

¿Qué es la programación visual?

Hasta el momento, la mayoría de los ejemplos vistos fueron aplicaciones de “consola”, es decir, el usuario interactúa con una interfaz de pantalla plana de caracteres, sin contenido gráfico. El tipo de programación que se aplica a estas interfaces es imperativa, ya que es el desarrollador quien decide de qué manera se iniciará, ejecutará y finalizará una aplicación.

Sin embargo, desde que surgieron los sistemas operativos con interfaz gráfica como Windows, Linux y macOS, la **interfaz gráfica de usuario (GUI)** pasó a ocupar la mayor parte de las aplicaciones de usuario final.

Este tipo de interfaz permite que el usuario interactúe con un programa de forma visual y además, el usuario final es quien decide qué hacer con una determinada aplicación o cualquier otra que tenga disponible.

En este tipo de interfaces, la programación es *por eventos*, es decir el desarrollador escribe el código para cada evento que produzca el usuario final.



Aplicaciones de escritorio Windows

.NET Framework cuenta con frameworks para interfaces gráficas: **WinForms** y **WPF (*Windows Presentation Foundation*)** para aplicaciones clientes de escritorio. Estos frameworks proveen un conjunto de componentes para “diseñar” formularios visuales.

La diferencia más importante entre **WinForms** y **WPF** es el hecho de que mientras **WinForms** es simplemente una capa sobre los controles estándar de Windows (por ejemplo, un **TextBox**), **WPF** está construido desde cero y casi no depende de los controles estándar de Windows.

En líneas generales, se puede decir que el componente principal, es el “*form*” (formulario), que cuenta con un encabezado. Allí están los botones de cerrar, maximizar y minimizar; un borde que permite mover y cambiar de tamaño el formulario; y un cuerpo (canvas) sobre el cual se van colocando los componentes que conformarán la interfaz visual propiamente dicha.



Aplicaciones Web Forms

Visual Studio .NET proporciona un entorno de programación visual orientado a objetos mediante el cual se pueden crear aplicaciones web utilizando componentes **ASP.NET** (por derivación y por composición). Esto permite desarrollar ese tipo de aplicaciones sin prestar mucha atención al HTML en sí, que no es más que el mecanismo a través del cual los distintos controles de nuestra interfaz se presentan al usuario final de nuestras aplicaciones.

Las aplicaciones web **ASP.NET** están formadas por formularios web que, usualmente, se dividen de la siguiente manera:

- un archivo con extensión **.aspx**, en el que se especifica la interfaz del formulario.
- un archivo con extensión **.aspx.cs** denominado *codebehind*, en el que se implementa la lógica de la aplicación.

A partir de la versión 2.0 de la plataforma **.NET**, que permite la implementación parcial de clases (con la palabra reservada `partial`), el archivo de código se divide en dos:

- en el archivo **.aspx.cs** el desarrollador implementa sus manejadores de eventos.
- un archivo **.aspx.designer.cs** en el que se agrupa todo el código generado automáticamente por **Visual Studio .Net**.

Para poder acceder a una aplicación web, es necesario publicarla en un servidor de aplicación **Internet Information Services** (a los fines de desarrollo **Visual Studio** proporciona un servidor local de **IIS**).

Al acceder a la página **.aspx**, el código se compila automáticamente y se genera un **assembly** en la caché del **CLR**. Si el contenido de la página cambia, el código se recompila automáticamente. Si no cambia, las solicitudes que se reciban a continuación utilizarán directamente la versión compilada que se halla en la caché, mejorando la eficiencia de las aplicaciones web **ASP.NET** con respecto a las versiones previas de **ASP**.



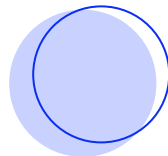
Controles web

Los controles web corresponden a las etiquetas **ASP.NET** `<asp:...>` y, como es lógico, también requieren el atributo **runat="server"** para funcionar, ya que estos elementos “corren en el servidor”.

La sintaxis de las etiquetas ASP.NET es la siguiente:

```
<asp:control id = "identificador" runat = "server" />
```

Donde `control` especifica el tipo de control web (etiquetas, botones, listas...) e `identificador` especifica el nombre de la variable mediante la cual se accederá al control en el código.



El desarrollo de aplicaciones web con controles **ASP.NET** es parecido al desarrollo de aplicaciones Windows. Sólo hay que seleccionar los controles adecuados para la interfaz e implementar la respuesta de la aplicación a los eventos que se desean controlar.



Control	Descripción
AdRotator	Muestra una secuencia de imágenes (p.ej. banners).
Button	Botón estándar.
Calendar	Calendario mensual.
CheckBox	Checkbox (como en los formularios Windows).
CheckBoxList	Grupo de checkboxes.
DataGrid	Grilla de datos.
DataList	Muestra una lista utilizando plantillas (templates).
DropDownList	Lista desplegable.
HyperLink	Enlace.
Image	Imagen.
ImageButton	Botón dibujado con una imagen.
Label	Etiqueta (texto estático).
LinkButton	Botón con forma de enlace.
ListBox	Lista (como en los formularios Windows).

Más controles

Control	Descripción
Literal	Texto estático (similar a Label).
Panel	Contenedor en el que se pueden colocar otros controles.
Placeholder	Espacio para controles agregados dinámicamente.
RadioButton	Botón de radio (como en los formularios Windows).
RadioButtonList	Grupo de botones de radio.
Repeater	Permite mostrar listas de controles "Data Binding".
Table	Tabla.
TextBox	Input de edición de texto.
Xml	Muestra un archivo XML o el resultado de una transformación XSL.

Controles

El IDE de **Visual Studio .NET** posee un “cuadro de herramientas” (**ToolBox**), que contiene una gran colección de componentes y/o controles (visuales y no visuales), agrupados por categorías.

La forma de trabajo es elegir uno o más componentes de esta barra y arrastrar y soltar el/los controles sobre el formulario; luego cada uno de estos tiene sus propiedades y eventos.

Los controles son lo esencial cuando se programa con formularios ya que sin estos lo único que podremos hacer será mostrar una ventana con controles de contenido textual y no tendríamos

posibilidad de permitirle al usuario una interacción con la aplicación. Los más utilizados:

- **Botón:** produce un evento cuando el usuario hace clic en el botón.
- **Etiqueta:** provee información en tiempo de ejecución o texto descriptivo para un control.
- **Caja de texto:** permite al usuario que escriba texto y provee edición multilínea y enmascara las palabras claves.

Todos estos y los demás controles (clases) tienen Propiedades, Métodos y Eventos.

Propiedades

Las propiedades son el equivalente a variables que tienen los controles para poder fijar sus características como tamaño, posición, color, fuente, etc. Estas propiedades se pueden ver a través del “Inspector de propiedades” que contiene el IDE.

Label

Propiedades

- **Text:** permite leer y escribir el contenido de texto de la etiqueta.



TextBox


Propiedades

- **MaxLength**: longitud de caracteres permitida.
- **TextMode**: *MultiLine*: permite ingresar varias líneas, *SingleLine* (valor predeterminado).
- **ReadOnly**: *True*: de sólo lectura, *False* (valor predeterminado).
- **TabIndex**: orden de secuencia de Tecla Tab.
- **Visible**: *False*: oculto, *True* (valor predeterminado).

Métodos

- **Focus()**: establece el foco al TextBox.

Eventos

- **TextChanged**: se produce cuando cambia el valor de la propiedad Text.
- 

Botón

Propiedades

- **CausesValidation:** *False*: no dispara la validación, *True* (valor predeterminado).
- **Enabled:** deshabilita o habilita el acceso al botón.

Eventos

- **Click:** se dispara al hacer clic el usuario.

DropDownList

Propiedades

- **SelectedItem:** opción seleccionada.

Eventos

- **SelectedIndexChanged:** se produce cuando cuando cambia la opción por el usuario.

CheckBox

Propiedades

- **Checked:** *True* = seleccionada, *False* (valor predeterminado).

Eventos

- **CheckedChanged:** se produce cuando cambia su valor.

RadioButtonList

Propiedades

- **SelectedValue:** valor seleccionado.

Eventos

- **TextChanged:** se produce cuando cambia de valor de texto del control.
- **SelectedIndexChanged:** se produce cuando cambia el valor del índice seleccionado.



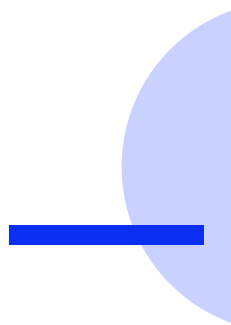
DateTimePicker

Propiedades

- **SelectedDate**: fecha seleccionada actualmente.

Eventos

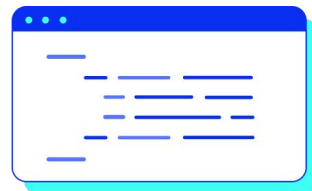
- **SelectionChanged**: se produce cuando el usuario cambia el valor seleccionado.



Eventos

Los eventos, son bloques de código, en general rutinas (procedimientos o funciones), que **contienen los controles para cada una de las posibles acciones que produzca el usuario** (hacer clic en un botón, mover el mouse, pulsar una tecla, escribir un texto, etc.).

El desarrollador codifica en el cuerpo de la rutina la acción que se ejecutará en caso de que se produzca ese evento. Los eventos también se ubican en la **“Ventana de Propiedades”** del IDE.



**¡Sigamos
trabajando!**