

Clase 17

EVENTOS

Programación y Laboratorio II

¿Qué son los **EVENTOS**?



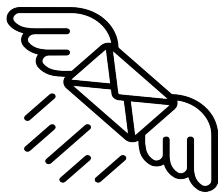
Un evento es un mensaje que envía un objeto cuando ocurre una **acción**.



Acción

La acción podría deberse a la interacción del usuario, como hacer clic en un botón, o podría derivarse de cualquier otra lógica del programa, como el cambio del valor de una propiedad.

Emisor receptor

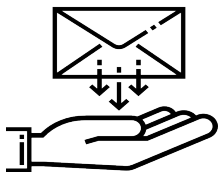


Emisor / remitente (sender)

Es el objeto que genera el evento.

Envía una notificación de que ha ocurrido un evento.

No sabe qué objeto o método recibirá (manejará) los eventos que genera.



Receptor

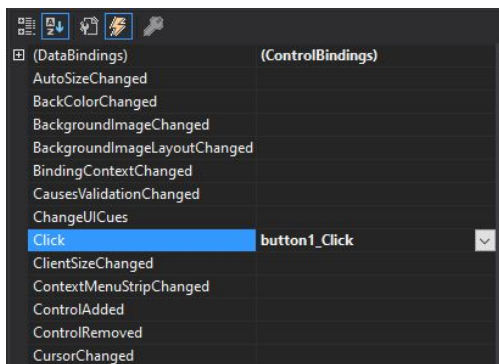
Recibe esa notificación y determina qué hacer.

Eventos en GUI



Cada **formulario** y **control** expone un conjunto de eventos. Si se produce uno de estos eventos y hay un código que lo controle, se invoca ese código. Muchos eventos ocurren junto con otros eventos.

*Cómo por ejemplo en el transcurso del evento de **DoubleClick**, se producen los eventos **MouseDown**, **MouseUp** y **Click**.*



```
1 private void botonEnviar_Click(object sender, EventArgs e)
2 {
3
4
5 }
```

Eventos y Delegados

El **delegado** es un intermediario entre el evento y el código que lo maneja.

Por convención, el identificador del delegado termina con la palabra **Handler** y el del evento empieza con la palabra **On**.

Declaración

Palabra reservada **event**

El tipo del evento debe ser un tipo de **delegado**

El identificador del evento debe ser un verbo o frase verbal

```
1  class Reloj
2  {
3      public delegate void NotificadorCambioTiempo(object reloj, InfoTiempoEventArgs infoTiempo);
4
5      public event NotificadorCambioTiempo SegundoCambiado;
6
7  }
```

Suscripción

El código que maneja el evento se lo conoce como **método manejador**.

Cuando se invoca el método, *FrmPrincipal_Load*, se asocia el evento de la clase *Reloj* con el método manejador.

El operador **+=** se utiliza para asociar sus manejadores con el evento.

```
1 public partial class FrmPrincipal : Form
2 {
3     private void FrmPrincipal_Load(object sender, EventArgs e)
4     {
5         Reloj reloj = new Reloj();
6         reloj.SegundoCambiado += MostrarCambioTiempo;
7     }
8
9     public void MostrarCambioTiempo(object reloj, InfoTiempoEventArgs info)
10    {
11        lblTiempo.Text = $"{info.hora}{info.minuto}{info.segundo}";
12    }
13 }
14 }
```

El operador **-=** permite desasociar el manejador al evento.

La firma del método manejador **debe** coincidir con la firma del delegado.

Invocar EVENTOS

```
1  if (SegundoCambiado is not null)
2  {
3      SegundoCambiado.Invoke(this, infoTiempo);
4  }
```

El evento es **null** si no tiene suscriptores.

El método *Invoke* se puede omitir.

El evento toma dos argumentos:

El origen del evento y el objeto derivado de *EventArgs*.

EventArgs

EventArgs es la clase base para todos los datos de eventos. Hereda todos sus métodos de *Object* y agrega un campo estático público llamado **Empty**, que representa un evento sin estado (para permitir el uso eficiente de eventos sin estado).

La clase EventArgs **se puede usar para proporcionar cualquier información sobre el evento.**

```
1 namespace System
2 {
3     //
4     // Resumen:
5     //     Representa la clase base para las clases que contienen datos de eventos y proporciona una
6     // valor a utilizar para eventos que no incluyen datos.
7     public class EventArgs
8     {
9         //
10        // Resumen:
11        //     Proporciona un valor para usar con eventos que no tienen datos de eventos.
12        public static readonly EventArgs Empty;
13
14        //
15        // Resumen:
16        //     Inicializa una nueva instancia de la clase System.EventArgs.
17        public EventArgs();
18    }
19 }
```

La clase suscriptor puede hacer coincidir fácilmente la firma del delegado requerida, simplemente tomando un parámetro de tipo *EventArgs*.

El suscriptor puede usar toda, parte o nada de la información pasada en EventArgs.



EJERCICIOS



Realizar los siguiente ejercicios:

- I01 - Avísame si cambia
- C01 - Temporizador



HTTP