UNIVERSIDAD AUTÓNOMA DE SINALOA

Facultad de Informática Culiacán

Introducción al Desarrollo De interfaces con WinForms en C#

Instructor:

MC. Gerardo Gálvez Gámez

gerardo.galvez@uas.edu.mx

Octubre de 2017

Temas

Introducción a las aplicaciones para Windows

1.1. Aplicaciones basadas en Windows de Visual Studio
1.2. Visual Studio Tools para el desarrollo de aplicaciones basadas en Windows
1.3. Clases de aplicaciones basadas en Windows
1.4. Aplicaciones de Windows Presentation Foundation
1.5. Aplicaciones de servicios de Windows
1.6. Proyectos Win32 y Win64
1.7. Servicios de aplicación cliente

Aplicaciones de Windows

- Son aquellas que:
  - Se ejecutan localmente en los equipos de los usuarios,
  - Obtienen acceso a los servicios del sistema operativo y
  - Su interfaz se diseña mediante formularios Windows Forms.
- Orientados a la optimización de recursos y procesos operativos del sistema Windows
- Aplicaciones empresariales y/o administrativas , que pueden trabajar en red local o con acceso remoto desde y hacia Internet

C# • FIUAS

Objetivo de aplicaciones Windows

- Automatizar tareas claves y repetitivas
- Diseñar rápidamente aplicaciones a la medida que permitan estar a un paso adelante de sus competidores
- Reducir el papeleo
- Hacer accesibles sus datos para una consulta más rápida desde un base de datos centralizada
- Configurar y dar soporte a sus sistemas computacionales
- Establecer servidores Windows que prácticamente trabajan por sí solos
- Fungir como su departamento de T.I. siempre disponibles

Conceptos clave en la programación en Windows.

- Ventanas.
  - Región de la pantalla.
    - Ventanas de documentos, botones, listas desplegables, cuadros de diálogo.
  - El SO administra todas las ventanas asignándolas un identificador.
- Eventos.
  - Acción que se ejecuta sobre el sistema.
  - El sistema operativo rastrea continuamente las ventanas en busca de sucesos.
- Mensajes.
  - Cuando se produce un evento se envía un mensaje al sistema operativo.
  - El mensaje guarda información sobre el suceso y la ventana que lo ha producido.
  - El sistema operativo lo registra y almacena en una cola de mensajes.

C# • FIUAS

Ejemplos de Aplicaciones de Windows

- Sistemas administrativos, ventas y facturación
- Sistemas de control de inventarios
- Sistemas de gestión de producción
- Sistemas de seguimientos y atención a clientes, etc.
- Entre otros.

Características de las aplicaciones Windows

- Existe una integración con el Sistema Operativo, tanto en el uso de recursos como en la parte visual.
- Es posible hacer uso de los recursos de CPU y almacenamiento de la PC donde se ejecuta la aplicación, por ejemplo tareas de CPU intensivas, guardar en cache datos comúnmente utilizados por la aplicación.
- Ofrece un mejor manejo y recuperación de errores que una aplicación de ASP.NET.

C# • FIUAS

Características de las aplicaciones Windows

- Es posible ejecutar ciertas tareas en otro hilo de ejecución (multi-thread), ofreciéndole al usuario la posibilidad de realizar varias actividades a la vez.
- La interface grafica es mas rica y funcional para el usuario quien cuenta con una serie de elementos que le ayudan a trabajar mejor.
- Es posible hacer que la aplicación trabaje de forma desconectada y que realice una sincronización de datos al momento de conectarse nuevamente a la red.

Desventajas

- Se requiere generalmente que la PC en que va a ejecutar la aplicación cuente con ciertos requisitos de hardware mínimos.
- Hay que instalar un mínimo de componentes en cada PC para que pueda funcionar la aplicación - como mínimo se requiere el .NET Framework -, aunque con características como ClickOnce es posible publicar la aplicación en un servidor de intranet y con un par de clicks el usuario mismo pueda auto instalar la aplicación.
- Cambios a la aplicación generalmente requiere de actualizar todos los clientes, pero es posible hacer que la aplicación misma busque por actualizaciones, notifique al usuario y se auto-actualice.

C# • FIUAS

Visual Studio Tools para el desarrollo de aplicaciones basadas en Windows

- Visual Studio .NET es que proporciona herramientas que hacen que el desarrollo de aplicaciones sea mucho más rápido, sencillo y confiable.

Herramientas

- Diseñadores visuales para formularios Windows Forms con controles de arrastrar y colocar.
- Editores de código inteligentes que incluyen finalización de instrucciones, comprobación de sintaxis y otras características de IntelliSense.
- Compilación y depuración integradas.
- Utilidades de administración de proyectos para la creación y administración de archivos de aplicación, incluida la implementación en servidores de intranet o Internet.

C# • FIUAS

Windows Presentation Foundation (WPF)

- Es un sistema de presentación de la próxima generación, para crear aplicaciones cliente de Windows que proporcionen una experiencia impactante para el usuario desde el punto de vista visual.
- Conjunto completo de características de desarrollo de aplicaciones que incluye Lenguaje XAML, controles, enlace de datos, diseño, gráficos 2D y 3D, animación, estilos, plantillas, documentos, multimedia, texto y tipografía.

Formularios Windows Forms

Aplicaciones Windows con C# .NET 2010

Temas

- Conceptos de programación en Windows
- Estructura de un procedimiento de evento
- Estructura de un proyecto Windows .NET

Conceptos de programación en Windows

- Ventana:
  - Región de la pantalla.
  - Ventanas de documentos, botones, listas desplegables, cuadros de diálogo.
  - El SO administra todas las ventanas asignándoles un identificador.
- Eventos:
  - Acción que se ejecuta sobre el sistema.
  - El sistema operativo rastrea continuamente las ventanas en busca de sucesos.
- Mensajes:
  - Cuando se produce un evento se envía un mensaje al sistema operativo.
  - El mensaje guarda información sobre el suceso y la ventana que lo ha producido.
  - El sistema operativo lo registra y almacena en una cola de mensajes.

C# • FIUAS

Conceptos de programación en Windows

- Programación orientada a eventos
  - El entorno (sistema operativo, usuario, etc.) puede actuar sobre el programa en cualquier momento.
  - El programa debe responder a las acciones del entorno no proporcionadas de forma lineal.
  - No se debe prever un desarrollo lineal del flujo del programa.
    - Las distintas acciones se activan como respuesta a sucesos que ocurren en el entorno.
- Al ejecutarse una aplicación basada en eventos
  - Windows rastrea las ventanas.
  - Si se detecta un evento en alguna ventana manda un mensaje al sistema operativo y lo almacena en la cola de mensajes
  - El sistema operativo lo procesa y lo transmite a las demás ventanas, indicando el evento y el identificador de la ventana que lo produce (Handle).
  - La aplicación busca el controlador de eventos asociado a ese evento en el control y, si existe, ejecuta el código correspondiente

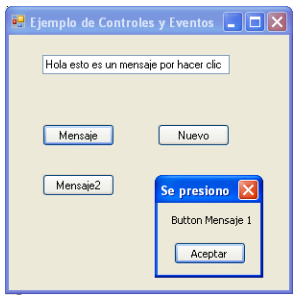
Estructura de un procedimiento de evento

- Cada componente de Windows Forms tiene asociado una serie de eventos a los que responde.
  - Los controladores de eventos tienen dos argumentos:
    - **Sender**, de tipo Object y tiene una referencia al objeto que lo ha producido.
    - **e**, un objeto de la clase EventArgs o alguna de sus derivadas con información del evento.
  - El nombre corresponde con el nombre del control.
  - La cláusula Handles indica que métodos de eventos están asociados al procedimiento.

Ejercicio

- Aplicación Windows que permite compartir eventos y distinguir que control fue el seleccionado:

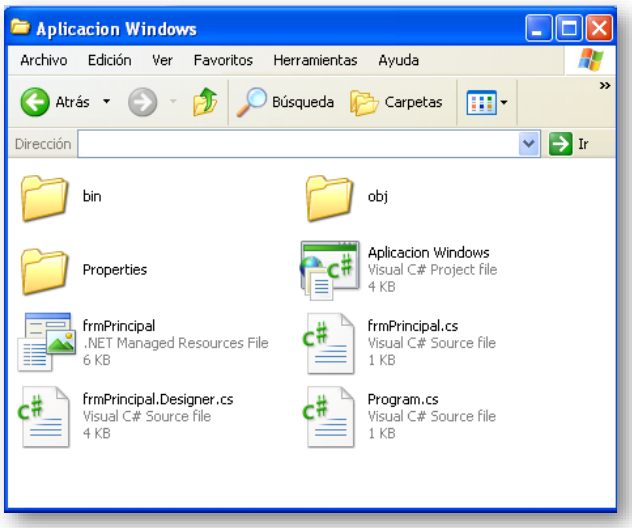
```
private void btnMensaje_Click(object sender, EventArgs e)
    {
      this.txtDato.Text = "Hola esto es un mensaje por hacer click en el boton";
      if (sender == btnMensaje)
      {
        MessageBox.Show("Button Mensaje 1","Se presiono");
      }
    }
```



Estructura de un proyecto Windows

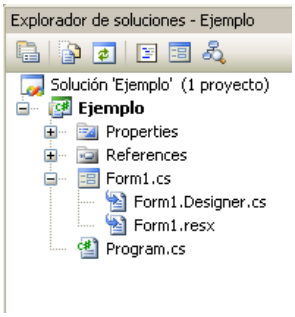
- Se desarrolla alrededor de uno o más formularios.
  - Generación automática de código.
- Visual Studio genera código en tres sitios distintos:
  - Archivo WindowsFormsApplicationX
    - Uno por proyecto
    - Está dentro del directorio del proyecto.
    - Incluye las características generales de la aplicación y formulario o módulo de arranque.
  - Archivo Form*X.Designer.cs*
    - Uno por formulario.
    - Dentro del directorio de proyecto.
    - Implementación parcial de la clase Form.
    - Incluye el código necesario para crear y destruir los controles que se incluyan en el formulario.
  - Archivo Form*X.cs.*
    - Clase Form*X con la declaración del resto de la clase.*
    - Incluye el código de usuario para manejar la aplicación.

Estructura de un proyecto Windows



La clase Form

- Representa una ventana o cuadro de diálogo de la aplicación.
  - Desde el punto de vista de la interfaz, se utilizará cómo un contenedor de controles.
  - Desde el punto de vista de la aplicación, será un objeto heredado de la clase Form y que constituye el punto de entrada de la aplicación.



- Normalmente contendrá las declaraciones y el código de la aplicación.
  - En el archivo Form*x.designer.cs*

Ciclo de vida de un formulario

- Eventos que intervienen.
  - 1. Evento Load().
    - Se produce cuando el formulario se carga por primera vez y antes de que se muestre.
    - Es el lugar adecuado para indicar el código necesario que permita inicializar variables, Indicar el contenido a los controles, etc.
  - 2. Evento Shown()
    - Se produce la primera vez que se muestra.
  - 3. Evento Activated().
    - Se produce cada vez que el formulario entra en foco, ya sea por una acción del usuario o por el código del programa.
    - Este evento se puede usar para la actualización del contenido con los cambios que pudieran haberse producido cuando no estaba activado.

Ciclo de vida de un formulario

- 4. Evento Deactivate().
  - Se produce cuando el formulario pierde el foco.
  - Puede utilizarse para actualizar el contenido de otra ventana con los datos del formulario que ha perdido el foco.
- 5. Evento FormClosing().
  - Se produce cuando se da la orden de cerrar el formulario, pero antes de que se cierre.
  - Es posible cancelar la acción de cierre poniendo a True la propiedad Cancel del argumento FormClosingEventArgs del control.
- 6. Evento FormClosed().
  - Se produce después de haberse cerrado el formulario.
  - Se puede utilizar para liberar recursos utilizados por el formulario, almacenar la información producida por él o actualizar otro formulario.

C# • FIUAS

Propiedades y Métodos

FORMS

Mover y cambiar el tamaño del formulario

- Métodos CenterToScreen() y CenterToParent().
  - Centran el formulario en la pantalla y en el formulario padre (en el caso de que sea una aplicación MDI).
- Propiedad TopMost.
  - Asignando un valor True, el formulario siempre aparece por encima del resto.
- Propiedad StartPosition.
  - Establece la posición de inicio del formulario.

Miembros de StartPosition	Decripción
CenterParent	El formulario está centrado en los límites de su formulario principal.
CenterScreen	El formulario está centrado en la pantalla actual y tiene las dimensiones especificadas en el tamaño del formulario.
Manual	La posición del formulario viene determinado por la propiedad Location
WindowsDefaultBounds	El formulario se encuentra colocado en la ubicación predeterminada de Windows y tiene los límites establecidos por Windows de forma predeterminada.
WindowsDefaultLocation	El formulario se encuentra colocado en la ubicación predeterminada de Windows y tiene las dimensiones especificadas en el tamaño del formulario

Modificar el aspecto del formulario

- Propiedad BackgroundImage.
  - Establece la imagen de fondo del formulario.
- Propiedad Icon.
  - Establece el icono de la barra de títulos del formulario.
- Propiedades ControlBox, MaximizeBox, MinimizeBox, HelpButton.
  - Contienen un valor lógico que establece si el botón del menú de control, maximizar, minimizar o el botón de ayuda aparecen en el formulario.
- Propiedad Opacity.
  - Establece mediante un número real el nivel de transparencia de un formulario.
  - De forma predeterminada el nivel de transparencia es de 100.
- Propiedad TransparencyKey.
  - Establece el color que será transparente en el formulario.
  - this.TranparencyKey = this.Backcolor 'Hace transparente el fondo del formulario.

Modificar el aspecto del formulario

- Propiedad FormBorderStyle.
  - Permite tomar alguno de estos valores:

Parámetro	Descripción
None	Ninguno (ningún borde ni elemento relacionado con él). Se utiliza para los formularios de inicio.
Fixed 3D	Se utiliza cuando se desea un efecto de borde tridimensional. No se puede cambiar de tamaño. Puede incluir en la barra de título un botón de menú de control y botones Maximizar y Minimizar.
Fixed Dialog	Se utiliza para los cuadros de diálogo. Presenta un borde grueso. No se puede cambiar de tamaño. Puede incluir en la barra de título un cuadro de menú de control, y botones Maximizar y Minimizar.
Fixed Single.	No se puede cambiar de tamaño. Presenta un borde de una sóla línea. Puede incluir cuadro de menú de control y botones Maximizar y minimizar. Sólo puede cambiar de tamaño con los botones Maximizar y Minimizar.
Fixed Tool Window	Se utiliza para las ventanas de herramientas. Muestra una ventana de tamaño no ajustable con un botón Cerrar y texto de barra de título con un tamaño de fuente reducido. El formulario no aparece en la barra de herramientas de Windows.
Sizable	Con frecuencia se utiliza como ventana principal. Se le puede cambiar el tamaño. Puede incluir un menú de control y botones Maximizar y Minimizar. Puede cambiar de tamaño mediante el cuadro de menú de control, los botones Maximizar y Minimizar de la barra de título, o mediante el ratón.
SizableToolWindow	Ventana de herramientas de tamaño variable. Una ventana de herramientas no aparece en la barra de tareas ni en la ventana que aparece cuando el usuario presiona ALT+TAB.

Control Form

Preguntar si el usuario quiere cerrar un Form

Evento ***FormClosing*** del control Form

```
//Código

if (MessageBox.Show("Cerrar la aplicación", "Titulo",
   MessageBoxButtons.YesNo,
   MessageBoxIcon.Question)==DialogResult.No)
{
      e.Cancel=true;
}
else
{
      e.Cancel=false;
}
```

Control Form

Mostrar un formulario como un cuadro de diálogo Modal.

```
1. Form2 ventana=new Form2();
2. ventana.ShowDialog();
```

Mostrar un formulario como un cuadro de diálogo No Modal.

```
1. Ventana.Show();
```

Invocar un Formulario hijo desde un formulario tipo MDIForm

```
Form1 Ventana=new Form1();
Ventana.MdiParent=this;
Ventana.Show();
```

Cerrar el Formulario

```
this.Close();
```

C#• FIUAS

Operaciones en Controles en Form

Pasar el focus al siguiente control en el formulario.

Se programa el evento Form_KeyPress, activando previamente la propiedad KeyPreview=True del formulario.

```
if ((Keys)e.KeyChar == Keys.Enter)
{
     SendKeys.Send("{TAB}");
    e.Handled=true; //cancelar la tecla
 }
```

Validaciones en un TextBox

- Solo se acepte números enteros
  - Se programa el evento KeyPress del control

Opción 1	Opción 2

if ((e.KeyChar>=48 && e.KeyChar<=58) || e.KeyChar<=32)
{
e.Handled=false;
}
else
{
e.Handled=true;
}
	string Numeros = "0123456789";
	if (Numeros.Contains(e.KeyChar)
	|| e.KeyChar<= 8)
	{
	e.Handled = false;
	}
	else
	{
	e.Handled = true;
	}

Validar numérico fraccionario

```
String  Numeros = "0123456789";
if (Numeros.Contains(e.KeyChar) || e.KeyChar==(char) Keys.Back )
{
        e.Handled = false; //acepta el carácter
}
else
{
   if (e.KeyChar=='.' && ! this.TextBox1.Text.Contains('.'))
    {
        e.Handled = false; // acepta el carácter
     }
    else
     {
        e.Handled = true;  //cancela el carácter
      }
}
```

Validación con RegularExpressions

using System.Text.RegularExpressions;

Programar el evento Validating del control

```
Regex Elementos = new Regex("[^0-9]");
if (Elementos.IsMatch(this.TextBox1.Text))
{
        e.Cancel = false;
 }
 else
{
                MessageBox.Show("Error, se requiere número entero");
                e.Cancel = true;
 }
```

C# • FIUAS

Ejemplo de RegularExpression

- **Para el caso monetario:**
  - **Regex Elementos = new Regex(@"^-?\d+(\.\d{2})?$");**
- Acepta números fraccionarios positivos o negativos:
  - ^[-+]?\d+(\.\d+)?$
- **Sitios relacionados:**
  - http://es.wikipedia.org/wiki/Expresi%C3%B3n_regular
  - http://regexlib.com/Search.aspx?k=decimal&c=-1&m=-1&ps=20

PictureBox

- Se suele utilizar el control PictureBox para mostrar gráficos de un archivo de mapa de bits, metarchivo, icono, JPEG, GIF o PNG.
- Asignar una imagen por código:
  - Bitmap MyImage = new Bitmap(FileName);
  - pictureBox.Image = (Image) MyImage ;
    - O
  - this.pictureBox.ImageLocation = FileName;

C# • FIUAS

Controles Avanzados

Ejemplo de operaciones comunes

RichTextBox

- Permite al usuario escribir y editar texto.
- Proporciona características de formato más avanzadas que el control TextBox estándar.
- El texto se puede asignar directamente al control o se puede cargar desde un archivo de formato de texto enriquecido (RTF) o de texto sin formato.
- Al texto del control se le puede asignar formato de carácter y párrafo.

OpenFileDialog

- Permite que los usuarios abran archivos mediante un cuadro de diálogo pre configurado.
- Ejecutar el cuadro de dialogo:

```
this.openFileDialog1.Filter = "archivo rtf|*.rtf";
this.openFileDialog1.ShowDialog();
```

- Programar el evento FileOk del control, parar recuperar el nombre del archivo seleccionado.
  - string RutaFileName= this.openFileDialog1.FileName;
  - Ejemplo : Abrir un archivo para el control **RichTextBox**

```
this.rtbEditor.LoadFile(this.openFileDialog1.FileName,
  RichTextBoxStreamType.RichText);
```

ColorDialog

- Es un cuadro de diálogo pre configurado que permite que el usuario seleccione un color de una paleta y agregue colores personalizados a la paleta. Es el mismo cuadro de diálogo que se ve en otras aplicaciones para Windows y que permite seleccionar colores.

- **Ejemplo:** Especificar un color para un texto seleccionado de un control **RichTextBox** :

```
if (colorDialog1.ShowDialog() == DialogResult.OK)
{
    this.RichTextBox1.SelectionColor = this.colorDialog1.Color;
}
```

C# • FIUAS

FontDialog

- Expone las fuentes actualmente instaladas en el sistema.

- Ejemplo: Llamar al control y asignarle la configuración seleccionada:

```
this.fontDialog1.ShowColor = true;
this.fontDialog1.Font = this.RichTextBox1.SelectionFont;
this.fontDialog1.Color = this.RichTextBox1.SelectionColor;
if (this.fontDialog1.ShowDialog() != DialogResult.Cancel)
{
    this.RichTextBox1.SelectionFont = this.fontDialog1.Font;
    this.RichTextBox1.SelectionColor =this.fontDialog1.Color;
}
```

SaveFileDialog

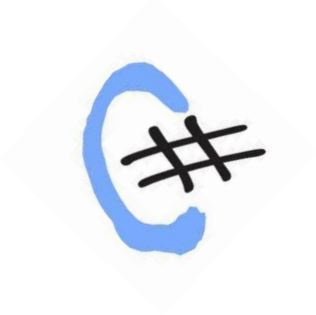
- Selecciona los archivos que se van a guardar y el lugar en el que se guardarán.
- Proceso de ejecución y grabación del contenido de un control **RichTextBox**:

```
this.saveFileDialog1.Filter = "archivos rtf | *.rtf";
this.saveFileDialog1.RestoreDirectory = true;
if (this.saveFileDialog1.ShowDialog() == DialogResult.OK)
{
    string  Archivo = this.saveFileDialog1.FileName;
    if (Archivo != string.Empty)
    {
          this. RichTextBox1.SaveFile(Archivo,
  RichTextBoxStreamType.RichText);
     }
 }
```

C# • FIUAS

Actividad

- Investigar y resumir el propósito del resto de los controles utilizados en el diseño de interfaces graficas.



Preguntas?