

TECNICATURA UNIVERSITARIA EN PROGRAMACIÓN

Programación III

Guía teórica práctica 1. Parte 2.

Autora: Tamara Gisele Herrera



Contenido

Controles RadioButton y CheckBox	2
Control GroupBox.....	3
Practica 6 (Radiobutton y CheckBox)	3
Control RichTextBox.....	5
Practica 7 (RichTextBox)	6
Controles ListBox y CheckedListBox.....	12
Practica 8 (ListBox y CheckedListBox)	13



Más controles de formulario

Controles RadioButton y CheckBox

Anteriormente comentamos que estos controles derivan de la misma clase Base que el control Button.

El control RadioButton se muestra como una etiqueta con un círculo, el mismo puede o no mostrarse por defecto como seleccionado. Estos controles RadioButton se utilizan para seleccionar entre una o más opciones de forma excluyentes Ej. Hombres o Mujeres. Para colocar un grupo de Controles RadioButton y que de forma automática siempre se seleccione una sola de las opciones, los mismos deben colocarse dentro de un GroupBox.

Un Control CheckBox se muestra como una etiqueta con una pequeña caja a su izquierda, estos controles CheckBox suelen utilizarse para seleccionar varias opciones, Ej. Un Cuestionario.

Propiedades más usadas en el Control RadioButton

PROPIEDADES	DESCRIPCIÓN
Appearance	<i>Como se muestra el control, con el circulo a la izq. o derecha.</i>
AutoCheck	<i>Indica si al seleccionarlo se marca de forma automática.</i>
CheckAlign	<i>Indica la alineación de la casilla dentro del control.</i>
Checked	<i>Indica si el control esta activado o no. (True o False)</i>

Eventos más usados en el Control RadioButton

EVENTOS	DESCRIPCIÓN
CheckedChanged	<i>Se dispara cuando hay cambios en el control RadioButton.</i>
Click	<i>Se dispara cuando se hace click sobre el control RadioButton, no es lo mismo que CheckedChanged ya que varios click solo cambia el estado una sola vez y CheckedChanged no vuelve a ser comprobado.</i>



Propiedades más usadas en el Control CheckBox

PROPIEDADES	DESCRIPCIÓN
CheckState	<i>Indica el estado, Chequeado, no Chequeado e Indeterminado.</i>
ThreeState	<i>Indica si el control permite tres estados o solo dos.</i>

Eventos más usados en el Control CheckBox

EVENTOS	DESCRIPCIÓN
CheckedChanged	<i>Se dispara cuando hay cambios en el control CheckBox.</i>
CheckStateChanged	<i>Se dispara cuando produce un cambio en la propiedad CheckState.</i>

Control GroupBox

El control GroupBox se utiliza a menudo para agrupar lógicamente un conjunto de controles tales como el RadioButton y CheckBox, y para proporcionar un título y un marco alrededor de este conjunto. Tenga en cuenta que si coloca la propiedad Enabled en false del Control GroupBox, todos los controles que estén dentro de él se deshabilitarán.

Practica 6 (Radiobutton y CheckBox)

Generar un Formulario con los controles y diseño que se muestran en la siguiente imagen:



Cambiar las Propiedades Name y Text de todos los Controles.

Los RadioButtons están dentro de un control GroupBox.

Cambiar la Propiedad CheckAlign.

Cambiar la Propiedad Checked del CheckBox Programador, para activarlo. Cambiar la Propiedad Checked del RadioButton Hombre, para activarlo.

Crear un método ValidarOk.

```
private void ValidarOk()
{
    //Habilita el Boton, siempre y cuando txtNombre tenga datos.
    btnOk.Enabled = (txtNombre.BackColor != Color.Red);
}
```

Manejamos el Evento Validating del Control TextBox Nombre.

```
private void txtNombre_Validating(object sender, CancelEventArgs e)
{
    TextBox tb = (TextBox)sender;
    if (tb.Text.Length == 0)
        tb.BackColor = Color.Red;
    else
        tb.BackColor = System.Drawing.SystemColors.Window;
    ValidarOk();
}
```

Manejamos el Evento Load del Formulario.

```
private void fmrRbtnCb_Load(object sender, EventArgs e)
{
    //Deshabilito el Boton Ok.
    btnOk.Enabled = false;
}
```



Manejamos el Evento Click del Boton Ok.

```
private void btnOk_Click(object sender, EventArgs e)
{
    //No valido datos ya que si el Boton esta Habilitado
    //es porque paso el Evento Validating del Nombre.
    String salida; //Declaro una variable para armar la salida.
    salida = "Nombre: " + txtNombre.Text + "\r\n";
    salida += "Ocupacion: " + (string)(cbxProgramer.Checked ?
        "Programador" : "No es Programador") + "\r\n";
    salida += "Sexo: " + (string)(rbtnHombre.Checked ? "Hombre" :
        "Mujer") + "\r\n";
    //Vuelco la salida al TextBox Salida.
    txtSalida.Text = salida;
}
```

El operador condicional ? evalúa una expresión si es true la primera expresión que sigue a ? se convierte en el resultado, si el false la segunda expresión se convierte en el resultado.

Control RichTextBox

El Control RichTextBox deriva de la clase TextBoxBase al igual que el control TextBox, por esto mismo es que comparten muchas de las propiedades y eventos.

A diferencia del control TextBox el RichTextBox soporta Texto con formato (Negrita, subrayado, cursiva), esto se logra utilizando un estándar conocido como Texto con Formato Enriquecido o RTF.

Posee más propiedades que el Control TextBox, estas propiedades que se agregan tienen que ver más con la selección del texto, ya que normalmente cuando uno da formato a un texto se lo hace por partes y en forma diferente.

Si uno no realiza una selección y aplica un cambio de formato este comenzara a aplicarse desde donde quedo el cursor.



Las Propiedades más utilizadas son

PROPIEDADES	DESCRIPCIÓN
CanRedo	<i>True cuando la última operación deshecha puede aplicar rehacer.</i>
CanUndo	<i>True si es posible deshacer la última acción en el control. Cuando se define en TextBoxBase, por lo que está disponible para los controles TextBox también.</i>
RedoActionName	<i>Contiene el nombre de una acción que se llevara a cabo con el método rehacer.</i>
DetectUrls	<i>Se coloca en true para detectar url y darle formato.</i>
Rtf	<i>Igual a text, solo que esta contiene el Texto en formato RTF.</i>
SelectedRtf	<i>Se utiliza para mantener el texto seleccionado en el control en formato RTF. así al copiar no pierde el formato.</i>
SelectedText	<i>Igual a SelectedRtf pero se pierde el formato.</i>
SelectionAlignment	<i>Representa la alineación del Texto Seleccionado.</i>
SelectionColor	<i>Color del Texto seleccionado.</i>
SelectionFont	<i>Cambia la fuente del Texto en la selección.</i>
SelectionLength	<i>Establece o recupera la longitud de la selección.</i>
SelectionType	<i>Contiene información de la selección.</i>
ShowSelectionMargin	<i>En True deja un margen a la izq. Para que sea más fácil seleccionar el texto.</i>
UndoActionName	<i>Obtiene el nombre de la acción que se utilizara si se opta por deshacer algo.</i>

Eventos más usados en el Control RichTextBox

EVENTOS	DESCRIPCIÓN
LinkClicked	<i>Se dispara cuando se hace Click en un Enlace dentro del Texto.</i>
Protected	<i>Se dispara cuando se intenta modificar un texto que se marcó como protegido.</i>
SelectionChanged	<i>Se dispara cuando se cambia la selección, si no quiero cambiar la selección puedo parar el cambio aquí.</i>

Practica 7 (RichTextBox)

Generar un Formulario con los controles y diseño que se muestran en la siguiente



imagen:



Cambiar las Propiedades Name y Text de todos los controles. Cambiar la Propiedad MaxLength del TextBox Tamaño de Fuente.

Cambiar la Propiedad Anchor de todos los controles con los siguientes valores:

CONTROL	VALOR
Guardar y Abrir	<i>Bottom.</i>
RichTextBox	Top, Left, Bottom, Right
Otros	<i>Top.</i>

Cambiar la Propiedad MinimumSize del Formulario.

Abrir el Evento Click del Boton Negrita y colocar el siguiente código.

```
private void btnNegrita_Click(object sender, EventArgs e)
{
    Font viejaFuente; //Declaro una variable Class Font para la Fuente Vieja.
    Font nuevaFuente; //Declaro una variable Class Font para la Fuente Nueva.
    //Asigno a viejaFuente el tipo de fuente seleccionado.
```




```
//Obtengo el tipo de Fuente atravez de la Propiedad SelectionFont
//del Control RichTextBox.
viejaFuente = rtxtEditor.SelectionFont;
if (viejaFuente.Bold) //Pregunto si ya es Negrita.
    //Si ya Tenia Negrita, establece nuevaFuente sin Negrita.
    nuevaFuente = new Font(viejaFuente, viejaFuente.Style & ~FontStyle.Bold);
else
    //Si no Tenia Negrita, establece nuevaFuente con Negrita.
    nuevaFuente = new Font(viejaFuente, viejaFuente.Style | FontStyle.Bold);
//Establesco la nuevaFuente a la Selección.
rtxtEditor.SelectionFont = nuevaFuente;
//Realizo un foco para posicionarme en el Control.
rtxtEditor.Focus();
}
```

Font Inicializa un nuevo objeto Font que utiliza el objeto Font existente especificado y la enumeración FontStyle(Negrita, Italic etc.).Font(Font, FontStyle) más Info en:

<https://learn.microsoft.com/es-es/dotnet/api/?view=netframework-4.5>

Abrir el Evento Click del Boton Subrayado y colocar el siguiente Código.

```
private void btnSubrayado_Click(object sender, EventArgs e) {
    Font viejaFuente; //Declaro una variable Class Font para la Fuente Vieja.
    Font nuevaFuente; //Declaro una variable Class Font para la Fuente Nueva.
    //Asigno a viejaFuente el tipo de fuente seleccionado.
    //Obtengo el tipo de Fuente atravez de la Propiedad SelectionFont
    //del Control RichTextBox.
    viejaFuente = rtxtEditor.SelectionFont;
    if (viejaFuente.Underline) //Pregunto si ya es Subrayado.
        //Si ya Tenia Subrayado, establece nuevaFuente sin Subrayado.
        nuevaFuente = new Font(viejaFuente, viejaFuente.Style & ~FontStyle.Underline);
    else
        //Si no Tenia Subrayado, establece nuevaFuente con Subrayado.
        nuevaFuente = new Font(viejaFuente, viejaFuente.Style | FontStyle.Underline);
    //Establesco la nuevaFuente a la Selección.
    rtxtEditor.SelectionFont = nuevaFuente;
    //Realizo un foco para posicionarme en el Control.
    rtxtEditor.Focus();
}
```



Abrir el Evento Click del Boton Cursiva y colocar el siguiente código.

```
private void btnCursiva_Click(object sender, EventArgs e) {
    Font viejaFuente; //Declaro una variable Class Font para la Fuente Vieja.
    Font nuevaFuente; //Declaro una variable Class Font para la Fuente Nueva.
    //Asigno a viejaFuente el tipo de fuente seleccionado.
    //Obtengo el tipo de Fuente atravez de la Propiedad SelectionFont
    //del Control RichTextBox.
    viejaFuente = rtxtEditor.SelectionFont;
    if (viejaFuente.Italic) //Pregunto si ya es Cursiva.
        //Si ya Tenia Cursiva, establece nuevaFuente sin Cursiva.
        nuevaFuente = new Font(viejaFuente, viejaFuente.Style & ~FontStyle.Italic);
    else
        //Si no Tenia Cursiva, establece nuevaFuente con Cursiva.
        nuevaFuente = new Font(viejaFuente, viejaFuente.Style | FontStyle.Italic);
    //Establesco la nuevaFuente a la Selección.
    rtxtEditor.SelectionFont = nuevaFuente;
    //Realizo un foco para posicionarme en el Control.
    rtxtEditor.Focus();
}
```

Abrir el Evento Click del Boton Centrar y colocar el siguiente código.

```
private void btnCentrado_Click(object sender, EventArgs e) {
    //Atravez de la propiedad SelectionAlignment, obtiene la alineación
    //del texto que se selecciono.
    //Pregunto si es en centrado.
    if (rtxtEditor.SelectionAlignment == HorizontalAlignment.Center)
        //Si estaba Centrado lo alinea a la Izq.
        rtxtEditor.SelectionAlignment = HorizontalAlignment.Left;
    else
        //Si no estaba centrado lo centra.
        rtxtEditor.SelectionAlignment = HorizontalAlignment.Center;
    rtxtEditor.Focus();
}
```

Creo un método propio llamado AplicarTamanoText y coloco el siguiente código:

```
//Metodo propio Aplicar Tamaño de Texto
private void AplicarTamanoText(string textSize) {
```



```
//Declaro una Var. y le asigno el valor pasado, previa Conversion.
float nuevoTam = Convert.ToSingle(textSize);
//Declaro un Objeto FontFamily para el tipo de Fuente actual.
FontFamily actualFuenteFamily;
//Declaro un Objeto Font para la nueva Fuente.
Font nuevaFuente;
//De la selección actual del RichTextBox obtengo
//el tipo de familia de la fuente.
actualFuenteFamily = rtxtEditor.SelectionFont.FontFamily;
//New Font Instancio el objeto nuevaFuente segun
//tipo de FFuente y tamaño.
nuevaFuente = new Font(actualFuenteFamily, nuevoTam);
rtxtEditor.SelectionFont = nuevaFuente;
}
```

Abrir el Evento KeyPress del TextBox Tamaño y colocar el siguiente código.

```
private void txtTamanio_KeyPress(object sender, KeyPressEventArgs e)
{
    //Valido que si se presiono una tecla que no se un numero,
    //Backspace o enter, se controle el evento.
    if ((e.KeyChar < 48 || e.KeyChar > 57) && e.KeyChar != 8 && e.KeyChar != 13) e.Handled =
true;
    //Pregunto si es Enter.
    else if (e.KeyChar == 13)
    {
        //Si es Enter, pregunto si la cantidad de caracteres es mayor a 0.
        if (txtTamanio.Text.Length > 0)
            //Llamo al metodo Aplicar Tamaño al Texto.
            AplicarTamanioText(txtTamanio.Text);
        //Si es Enter y no se ingresaron caracteres, controlo el evento.
        e.Handled = true;
        rtxtEditor.Focus();
    }
}
```

Abrir el Evento Validated del TextBox Tamaño y colocar el siguiente código:

```
private void txtTamanio_Validated(object sender, EventArgs e) {
    //El Evento Validated ocurre despues de la validación(Evento Validating).
    //Llama al Metodo AplicarTamanioText y como parametro pasa,
    //La propiedad Text del objeto sender(Objeto que disparo el evento)
```



```
//Del cual se hace un Cast del tipo TextBox.  
//Es lo mismo que pasarle txtTamaño.text.  
AplicarTamañoText(((TextBox)sender).Text);  
rtxtEditor.Focus();  
}
```

Abrir el Evento LinkClicked del RichTextBox y Colocar lo siguiente:

```
private void rtxtEditor_LinkClicked(object sender, LinkClickedEventArgs e) {  
    //La clase Process brinda acceso a procesos locales y remotos.  
    //El metodo Start inicia un proceso y asocia el recurso a un nuevo componente.  
    System.Diagnostics.Process.Start(e.LinkText);  
}
```

Abrir el Evento Click del Boton Abrir y colocar el siguiente código.

```
private void btnAbrir_Click(object sender, EventArgs e) {  
    try  
    {  
        rtxtEditor.LoadFile("Test.rtf");  
    }  
  
    catch (System.IO.FileNotFoundException)  
    {  
        MessageBox.Show("No se pudo cargar el archivo");  
    }  
}
```

Abrir el Evento Click del Boton Guardar y colocar el siguiente código.

```
private void btnGuardar_Click(object sender, EventArgs e) {  
    try  
    {  
        rtxtEditor.SaveFile("Test.rtf");  
    }  
    catch (System.Exception err)  
    {  
        MessageBox.Show(err.Message);  
    }  
}
```



Controles ListBox y CheckedListBox

Los ListBox se usan para mostrar una lista de cadenas o valores donde el usuario puede seleccionar uno o más a la vez.

La clase ListBox se deriva de la clase ListControl que proporciona la funcionalidad básica para todos los controles de cuadro de lista que proporciona el .NET Framework.

Otro tipo de cuadro de lista disponible se llama CheckedListBox que deriva de la clase ListBox, que proporciona una lista al igual que el ListBox, pero además de las cadenas de texto proporciona una casilla de verificación para cada elemento en la lista.

Propiedades comunes para ListBox y CheckedListBox

PROPIEDADES	DESCRIPCIÓN
SelectedIndex	<i>Indica el índice seleccionado de la lista. Si es una selección múltiple, devuelve el primer índice seleccionado.</i>
ColumnWidth	<i>Especifica el ancho de columna en un cuadro de lista con varias columnas.</i>
Items	<i>Devuelve la colección de ítems de todos los elementos en el cuadro de lista. Se pueden agregar o quitar elementos de esta colección.</i>
MultiColumn	<i>Un cuadro de lista puede tener más de una columna, esta propiedad especifica si el valor se muestra en cantidades de columnas.</i>
SelectedIndices	<i>Devuelve una colección con todos los índices de los elementos seleccionados.</i>
SelectedItem	<i>Devuelve el elemento seleccionado, si son varios devuelve el primero.</i>
SelectedItems	<i>Devuelve una colección con todos los elementos seleccionados.</i>
SelectionMode	<i>Modo de selección existen 4 modos.</i>
Sorted	<i>En true ordena los elementos de la lista</i>
Text	<i>Se selecciona en el list el elemento que coincida con el Text.</i>
CheckedIndices	<i>Devuelve una colección con todos los índices de los elementos del CheckedListBox que tengan un estado chequeado.</i>
CheckedItems	<i>Devuelve una colección con todos los elementos del CheckedListBox que tengan un estado chequeado.</i>
CheckOnClick	<i>(Solo CheckedListBox) En true el estado del control cambia con un click.</i>
ThreeCheckBoxes	<i>Cambia el estilo de la casilla de verificación.</i>

Métodos comunes para ListBox y CheckedListBox

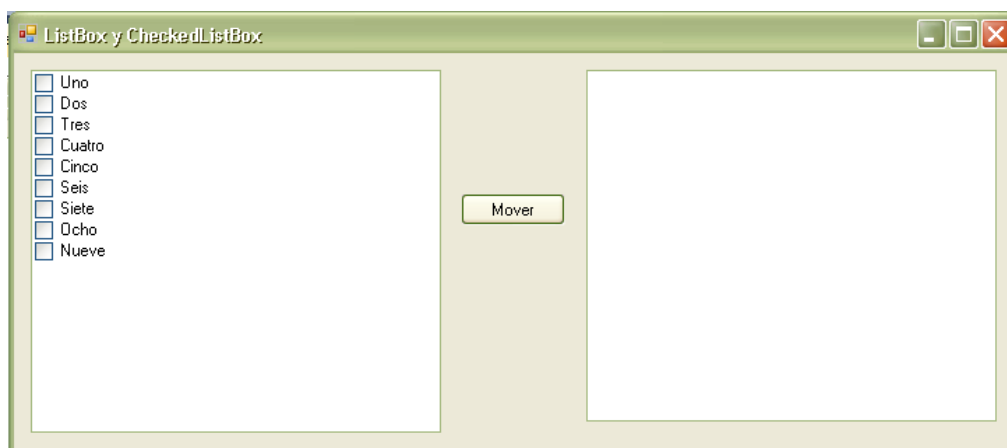
METODOS	DESCRIPCIÓN
ClearSelected()	<i>Limpia toda la selección del ListBox.</i>
FindString()	<i>Busca en el ListBox la primera cadena que en el comienzo coincida con la cadena que se especifica en este método.</i>
FindStringExact()	<i>Igual a la anterior pero la búsqueda es exacta.</i>
GetSelected()	<i>Devuelve un valor que indica si esta seleccionado.</i>
SetSelected()	<i>Establece o Borra la selección de un elemento.</i>
ToString()	<i>Devuelve el ítem seleccionado en formato String.</i>
GetItemChecked()	<i>(Solo CheckedListBox) Devuelve un valor si esta seleccionado.</i>
GetItemCheckState()	<i>(Solo CheckedListBox) Devuelve un valor que indica el estado de activación del ítem.</i>
SetItemChecked()	<i>(Solo CheckedListBox) Establece el elemento seleccionado a un estado seleccionado.</i>
SetItemCheckState()	<i>(Solo CheckedListBox) Establece el estado de activación de un elemento.</i>

Eventos comunes para ListBox y CheckedListBox:

EVENTOS	DESCRIPCIÓN
ItemCheck	<i>(Solo CheckedListBox) Ocurre cuando cambia el estado del check de un elemento.</i>
SelectedIndexChanged	<i>Se produce cuando hay cambios en los índices de los elementos seleccionados.</i>

Practica 8 (ListBox y CheckedListBox)

Ejercicio: Generar un Formulario con los controles y diseño que se muestran en la siguiente imagen:





Cambiar las Propiedades Name de todos los controles. Cambiar la Propiedad Text del Formulario y botón.

Cambiar la Propiedad CheckOnClick del CheckedListBox a True. Para que las casillas de verificación se activen con un Click.

Abrir el editor de Items del CheckedListBox, esto se hace desde la propiedad Items hacer click en Colección Luego Agregar Uno, Dos, Tres, Cuatro, Cinco, Seis, Siete, Ocho, Nueve.

Agregar un elemento más, pero desde el Código del formulario, en este caso en el constructor de la clase del formulario.

```
public frmListBoxCheckedListBox()  
{  
    InitializeComponent();  
    //Con el metodo Add agregamos un ultimo Item a la  
    //Colección de Items. Como parametro la cadena de caracteres.  
    clbValores.Items.Add("Diez");  
}
```

Agregar el evento Click del Boton Mover, para colocar el código que mueve al ListBox los Items seleccionados en el CheckedListBox.

```
private void btnMover_Click(object sender, EventArgs e)  
{  
    //Pregunta si la cantidad (Count) de elementos chequeados  
    //en la coleccion de items chequeados(CheckedItems) es mayor a cero.  
    if (clbValores.CheckedItems.Count > 0)  
    {  
        //Si es mayor a cero. Limpia los Item en el ListBox.  
        lbSeleccionados.Items.Clear();  
        //La sentencia foreach recorre la coleccion de Items Seleccionados  
  
        //Y los asigna de a uno a la variable item del tipo String.  
        foreach (string item in clbValores.CheckedItems)  
        {  
            //Agrego los items seleccionados en la colección  
            //Al Listbox con el Metodo Add.  
            lbSeleccionados.Items.Add(item.ToString());  
        }  
    }  
}
```



```
//Recorro todos los Items del CheckedListBox.  
for (int i = 0; i < clbValores.Items.Count; i++)  
    //Con el Metodo SetItemChecked, establezco en falso la  
    //casilla de verificación (No esta seleccionado).  
    //Como parametros i-El Indice y el valor de estado en este caso  
    false.  
clbValores.SetItemChecked(i, false);  
}
```

Para más información sobre algunos métodos la pueden encontrar en
<https://learn.microsoft.com/es-es/dotnet/api/?view=netframework-4.5>