# Win32 API (Application Programming Interface).

**PROGRAMACIÓN 2**

MGTI ALEJANDRO VILLARREAL MOJICA

# GetDlgItem

- La función **GetDlgItem** devuelve el manipulador de un control (Variable HWND – Windows Handle) en el cuadro de diálogo especificado.

```
HWND GetDlgItem(
    HWND hDlg,      // manipulador del cuadro de diálogo
    int nIDDlgItem  // identificador del control
   );
```

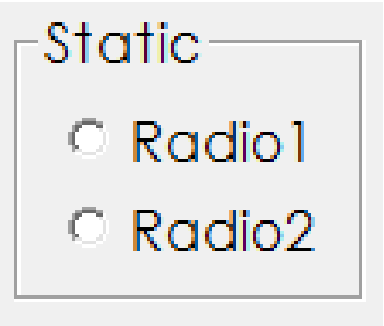
# GetDlgItem

- Parámetros:
  - **hDlg:** identifica el cuadro de diálogo que contiene el control.
  - **nIDDlgItem:** especifica el identificador del control del que se quiere recuperar el manipulador.

- Valor de retorno:
  - Si la función tiene éxito, el valor de retorno es el manipulador de la ventana del control.
  - Si la función falla el valor de retorno en NULL, que indica que el manipulador de diálogo no es válido o que el control no existe.

# RADIO BUTTON GroupBox

- Los RadioButtons sólo pueden tomar dos valores, encendido y apagado. Se usan típicamente para leer opciones que sólo tienen un número limitado y pequeño de posibilidades y sólo un valor posible, como por ejemplo: sexo (hombre/mujer), estado civil (soltero/casado/viudo/divorciado), etc.



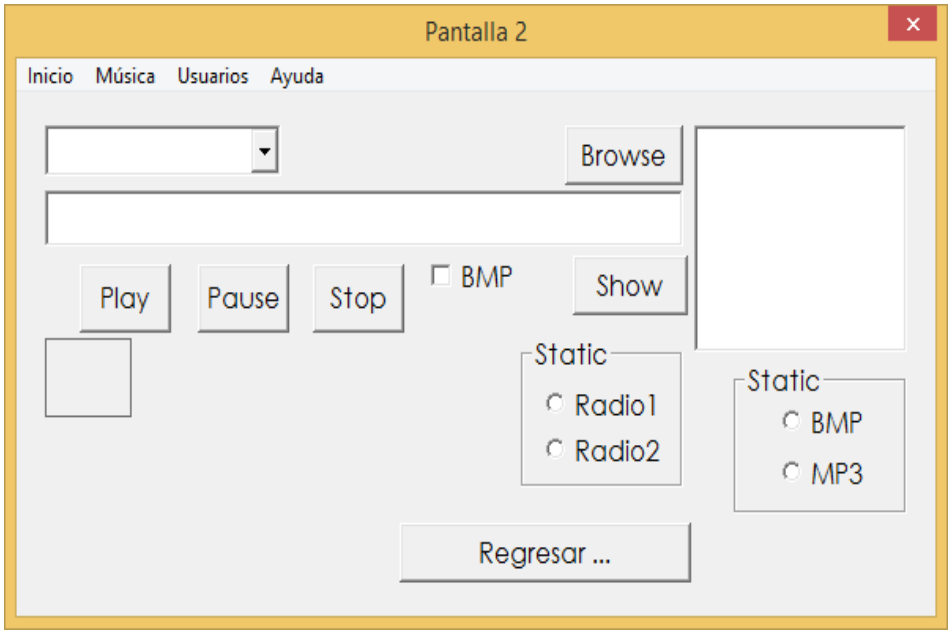

- Es necesario agrupar usando un GroupBox:
  - Los GroupBoxes son un estilo de botón que se usa para agrupar controles. Generalmente se usan con controles RadioButton, pero se pueden agrupar controles de cualquier tipo.
  - El comportamiento es puramente estático, es decir, actúan sólo como marcas y facilitan al usuario el acceso a distintos grupos de controles asociados en función de alguna propiedad común.

# RADIO BUTTON

- El aspecto normal es el de un pequeño círculo con un texto a uno de los lados, normalmente a la derecha.
- Cuando está activo se muestra el círculo relleno, cuando no lo está, aparece vacío. También es posible mostrar el RadioButton como un botón corriente, en ese caso, al activarse se quedará pulsado.

# RADIO BUTTON

- No tiene sentido colocar un solo control RadioButton, ya que al menos uno de cada grupo debe estar activo.
- Tampoco es frecuente agrupar dos, ya que para eso se puede usar un único control CheckBox. Tampoco se agrupan demasiados, ya que ocupan mucho espacio, en esos casos es mejor usar un ComboBox o un ListBox.



# RADIO BUTTON Inicializarlos

- Para establecer los valores iniciales de los controles RadioButton usaremos el mensaje WM_INITALOG del procedimiento de diálogo.
- Para eso usaremos la función CheckRadioButton o el mensaje BM_SETCHECK, en este último caso, emplearemos la función SendDlgItemMessage o SendMessage.
- Usar el mensaje implica enviar un mensaje al menos a dos controles RadioButton del grupo, el que se activa y el que se desactiva.

# RADIO BUTTON No Automáticos

Siempre y cuando los objetos
ID_RADIOBUTTON4
ID_RADIOBUTTON5
ID_RADIOBUTTON6
Están dentro de un mismo grupo, y hayan sido creados secuencialmente, es decir, con los ID's de forma consecutiva.

```
switch(LOWORD(wParam)) {
        case ID_RADIOBUTTON4:
        case ID_RADIOBUTTON5:
        case ID_RADIOBUTTON6:
            CheckRadioButton(hDlg,
        ID_RADIOBUTTON4,
        ID_RADIOBUTTON6, LOWORD(wParam));
          return TRUE;
```

NOTA:
Se aplica esto siempre y cuando los RadioButton's la propiedad Auto = False.

# RADIO BUTTON No Automáticos

```
// Similar a la diapositiva anterior, pero utilizando la función de Enviar Mensaje:
switch(LOWORD(wParam)) {
        case ID_RADIOBUTTON4:
        case ID_RADIOBUTTON5:
        case ID_RADIOBUTTON6:
          SendDlgItemMessage(hDlg, ID_RADIOBUTTON4, BM_SETCHECK,  BST_UNCHECKED,
0);
          SendDlgItemMessage(hDlg, ID_RADIOBUTTON5, BM_SETCHECK,  BST_UNCHECKED,
0);
          SendDlgItemMessage(hDlg, ID_RADIOBUTTON6, BM_SETCHECK,  BST_UNCHECKED,
0);
          SendDlgItemMessage(hDlg, LOWORD(wParam), BM_SETCHECK, BST_CHECKED, 0);
          return TRUE;
```

NOTA: Se aplica esto siempre y cuando los RadioButton's la propiedad Auto = False.

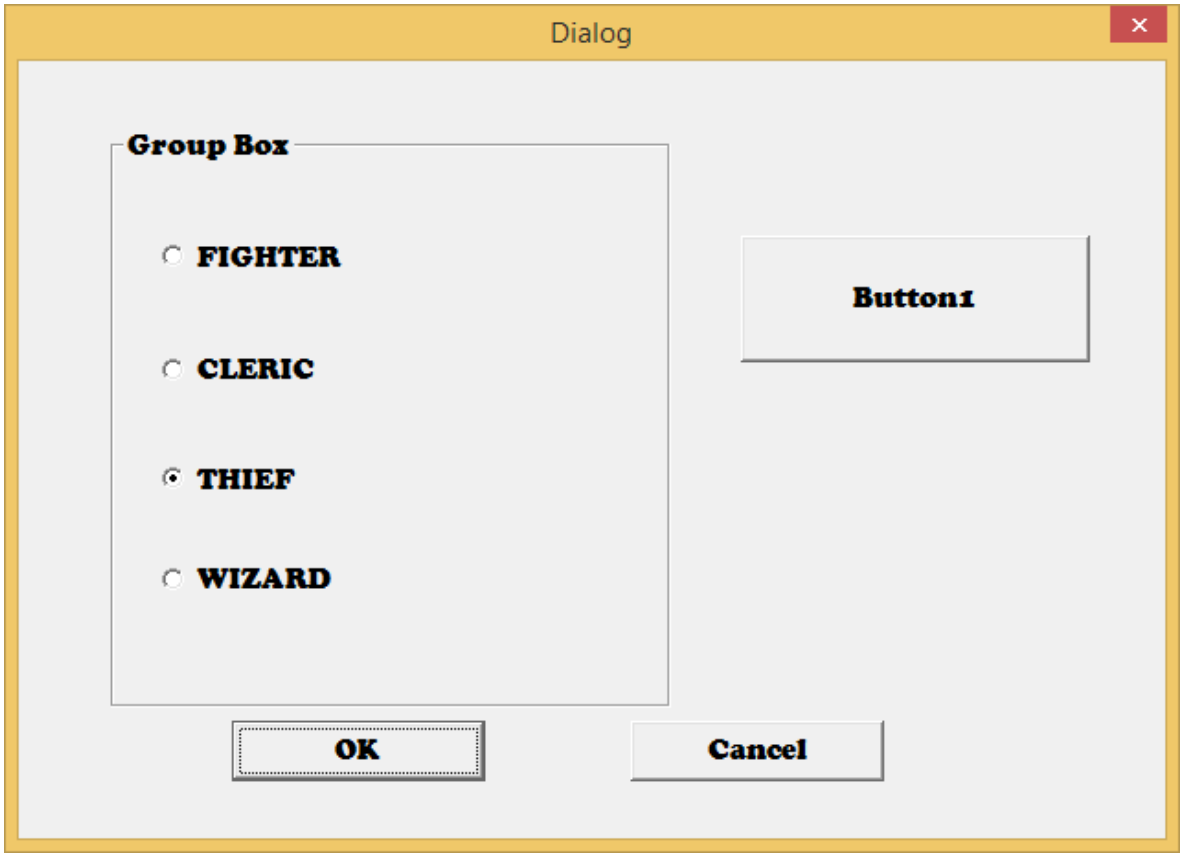
# RadioButton Leer el valor

```
if(IsDlgButtonChecked(hDlg, ID_RADIOBUTTON1) == BST_CHECKED)
if(IsDlgButtonChecked(hDlg, ID_RADIOBUTTON2) == BST_CHECKED)

if(SendDlgItemMessage(hDlg, ID_RADIOBUTTON4,
                      BM_GETCHECK, 0, 0) == BST_CHECKED)
if(SendDlgItemMessage(hDlg, ID_RADIOBUTTON5,
                      BM_GETCHECK, 0, 0) == BST_CHECKED)
```

# Radio Buttons

Se ***DEBE*** elegir una opción.




MGTI Alejandro Villarreal --> alvillarreal@gmail.com

# Radio Button

```
switch( msg )
{
case WM_INITDIALOG:
// Select the default radio button.
// Note: Assumes the radio buttons were created
// sequentially.

CheckRadioButton(hDlg,
IDC_RADIO_FIGHTER,    // First radio button in group
IDC_RADIO_WIZARD, // Last radio button in group
IDC_RADIO_THIEF);     // Button to select.

classSelection = IDC_RADIO_THIEF;
```

# Radio Button

```
case WM_COMMAND:
switch(LOWORD(wParam))
{
case IDC_RADIO_FIGHTER:       // Was *any* radio button selected?
case IDC_RADIO_CLERIC:
case IDC_RADIO_THIEF:
case IDC_RADIO_WIZARD:
// Yes, one of the radio buttons in the group was selected,
// so select the new one (stored in LOWORD(wParam)) and
// deselect the other to update the radio button GUI.
// Note: Assumes the radio buttons were created sequentially.
CheckRadioButton(hDlg,
    IDC_RADIO_FIGHTER,// First radio button in group
    IDC_RADIO_WIZARD, // Last radio button in group
    LOWORD(wParam)); // Button to select.
// Save currently selected radio button.
classSelection = LOWORD(wParam);
```

MGTI Alejandro Villarreal --> alvillarreal@gmail.com