# 这一节我们来学习按钮类控件

# 学习大纲

|  |
| --- |
|  |

## IsWindowVisible函数的语法

|  |
| --- |
| 确定指定窗口的可见性状态。 语法 BOOL IsWindowVisible(  [in] HWND hWnd  ); 参数 [in] hWnd  类型：**HWND**  要测试的窗口的句柄。 返回值 类型： **BOOL**  如果指定的窗口、其父窗口、其父窗口等具有 **WS\_VISIBLE** 样式，则返回值为非零。 否则返回值为零。  由于返回值指定窗口是否具有 **WS\_VISIBLE** 样式，因此即使窗口被其他窗口完全遮挡，也可能为非零。 注解 窗口的可见性状态由 **WS\_VISIBLE** 样式位指示。 设置 **WS\_VISIBLE** 后，只要窗口具有 **WS\_VISIBLE** 样式，就会显示窗口的后续绘制。  如果窗口被其他窗口遮盖或被其父窗口剪裁，则不会显示具有 **WS\_VISIBLE** 样式的窗口的任何绘图。 |

## EnableWindow函数的语法

|  |
| --- |
| BOOL EnableWindow(  [in] HWND hWnd,  [in] BOOL bEnable  ); 参数 [in] hWnd  类型：**HWND**  要启用或禁用的窗口的句柄。  [in] bEnable  类型： **BOOL**  指示是启用或禁用窗口。 如果此参数为 **TRUE**，则启用窗口。 如果 参数为 **FALSE**，则禁用窗口。 返回值 类型： **BOOL**  如果以前禁用窗口，则返回值为非零值。  如果以前未禁用窗口，则返回值为零。 注解 如果窗口处于禁用状态，系统会发送 [WM\_CANCELMODE](https://learn.microsoft.com/zh-cn/windows/desktop/winmsg/wm-cancelmode) 消息。 如果窗口的启用状态正在更改，系统会在 [WM\_CANCELMODE](https://learn.microsoft.com/zh-cn/windows/desktop/winmsg/wm-enable) 消息后发送 **WM\_ENABLE** 消息。 (这些消息在 **EnableWindow** 返回之前发送。) 如果窗口已禁用，则其子窗口将被隐式禁用，尽管它们不会发送 **WM\_ENABLE** 消息。  必须先启用窗口才能激活它。 例如，如果应用程序显示无模式对话框并禁用其main窗口，则应用程序必须在销毁对话框之前启用main窗口。 否则，另一个窗口将接收键盘焦点并被激活。 如果禁用了子窗口，当系统尝试确定哪个窗口应接收鼠标消息时，将忽略该窗口。  默认情况下，创建的窗口最初处于已启用状态。 若要创建最初禁用的窗口，应用程序可以在 [CreateWindow](https://learn.microsoft.com/zh-cn/windows/desktop/api/winuser/nf-winuser-createwindowa) 或 [CreateWindowEx](https://learn.microsoft.com/zh-cn/windows/desktop/api/winuser/nf-winuser-createwindowexa) 函数中指定**WS\_DISABLED**样式。 创建窗口后，应用程序可以使用 **EnableWindow** 启用或禁用窗口。  应用程序可以使用此函数来启用或禁用对话框中的控件。 禁用的控件无法接收键盘焦点，用户也无法获取对它的访问权限。 |

## IsWindowEnable函数的语法

|  |
| --- |
| 确定是否针对鼠标和键盘输入启用指定的窗口。 语法 C++复制  BOOL IsWindowEnabled(  [in] HWND hWnd  ); 参数 [in] hWnd  类型：**HWND**  要测试的窗口的句柄。 返回值 类型： **BOOL**  如果启用窗口，则返回值为非零值。  如果未启用窗口，则返回值为零。 注解 子窗口仅当同时启用且可见时才接收输入。 |

# 演练

## 1，新建一个常规的空项目，取名：Lesson28-buttondemo，然后新建一个cpp源文件，取名：buttondemo.cpp

|  |
| --- |
|  |

## 2.然后我们添加下面的代码

|  |
| --- |
|  |

## 3.然后我们利用对话框资源来创建对话框应用程序，在资源文件夹上面点击右键，添加-》资源->Dialog，就可以插入一个对话框

|  |
| --- |
|  |

## 4、我们先定义一个对话框窗口过程回调函数，然后创建一个对话框应用程序，保留确定按钮，把文字改为退出，把取消按钮删除，然后添加一些最基本的处理代码，例如，防止用户一按下回车键程序就结束，用户点击x按钮，程序需要退出等等

|  |
| --- |
| #include<Windows.h>  #include"resource.h"  INT\_PTR CALLBACK Dlgproc(  HWND hDlg,  UINT uMsg,  WPARAM wParam,  LPARAM lParam  )  {  BOOL ret = TRUE;  int cmdId;  switch(uMsg)  {  case WM\_COMMAND:  cmdId = LOWORD(wParam);  switch(cmdId)  {  case IDOK://防止用户按回车退出程序  if(IDOK == MessageBox(hDlg,TEXT("Exit App?"),TEXT("Info"),MB\_OKCANCEL))  {  EndDialog(hDlg,FALSE);  }  break;  }  break;  case WM\_CLOSE: //用户点击x按钮直接退出程序  EndDialog(hDlg,FALSE);  default:  ret = FALSE;  break;  }  return ret;  }  int APIENTRY WinMain(HINSTANCE hInstance,HINSTANCE prec,LPTSTR lpCmdLine,int nCmdShow)  {  DialogBox(hInstance,MAKEINTRESOURCE(IDD\_DIALOG1),NULL,Dlgproc);  return 0;  } |

## 5.我们在对话框中添加一个按钮，一个复选框和一个单选按钮，ID保持默认

|  |
| --- |
|  |

## 6.然后我们添加对他们点击的事件处理代码

|  |
| --- |
|  |

## 7.然后我们再添加3个按钮,第一个按钮用来设置按钮1的文本，第二个用来启用/禁用按钮1，第三个按钮用来勾选和取消勾选复选框函数单选按钮，第一个和第二个按钮的点击事件代码比较简单，第三个按钮需要使用SendMessage函数发送BM\_GETCHECK和BM\_SETCHECK消息来实现

|  |
| --- |
|  |
|  |

### 注意，用SendMessage函数发送勾选消息时，需要将wParam（也就是第三个参数）设置为1，用SendMessage函数发送取消勾选消息时，需要将wParam设置为0

## 8.然后我们添加一个按钮来获取复选框的选中状态，ID为IDC\_BTN\_GETSTATE，然后我们给他添加点击事件代码，注意获取选中状态需要定义一个变量来接收SendMessage函数的返回值，checked是非0，unchecked是0

|  |
| --- |
|  |

### 效果

|  |
| --- |
|  |
|  |

## 9.然后我们可以添加一个按钮来获取按钮1的文本，ID为：IDC\_BTN\_GETTEXT,然后我们给他添加点击事件处理代码

|  |
| --- |
|  |
|  |

## 10，然后我们再添加一个按钮，控制上面三个按钮是否可见，文本为Show/hide Buttons,ID为IDC\_BTN\_SHOW\_HIDE

|  |
| --- |
|  |

## 11,然后我们给这个按钮的点击事件添加代码，为了方便显示或者隐藏这三个按钮，我们把代码封装在一个叫做ToggleShowWin函数里面，然后在按钮事件里面调用这个函数

|  |
| --- |
|  |
|  |

### 效果

|  |
| --- |
|  |
|  |

## 12.根据类似的思想，我们也可以把启用/禁用按钮的处理代码封装为一个ToggleEnableCtrl函数，然后方便扩展这三个按钮的启用和禁用

|  |
| --- |
|  |
|  |

### 效果

|  |
| --- |
|  |
|  |

### 当然上面的代码也可以这么写，使用三目运算符

|  |
| --- |
|  |

### 还可以这么写，这里的技术要点是里面的IsWindowEnabled需要在前面添加！取反。

|  |
| --- |
|  |

### 同样道理，显示隐藏按钮的处理代码也是可以这么用的。

|  |
| --- |
|  |

### 效果是一样的

|  |
| --- |
|  |
|  |

### 这也是一种技巧，使用一个函数的返回值作为另外一个函数的一个参数，适当时候需要取反处理

# 这一节的学习到此为止，完整代码如下

## buttondemo.cpp

|  |
| --- |
| #include<Windows.h>  #include"resource.h"  HWND hwndBtn,hwndChk,hwndRadio;  void ToggleShowWin(HWND hwnd)  {  if(IsWindowVisible(hwnd))  {  ShowWindow(hwnd,SW\_HIDE);  }  else  {  ShowWindow(hwnd,SW\_SHOW);  }  }  void ToggleEnableCtrl(HWND hwnd)  {  if(IsWindowEnabled(hwnd))  {  EnableWindow(hwnd,FALSE);  }  else  {  EnableWindow(hwnd,TRUE);  }  }  INT\_PTR CALLBACK Dlgproc(  HWND hDlg,  UINT uMsg,  WPARAM wParam,  LPARAM lParam  )  {  static TCHAR szCaption[128];  BOOL ret = TRUE;  int cmdId;  hwndBtn = GetDlgItem(hDlg,IDC\_BUTTON1);  hwndChk = GetDlgItem(hDlg,IDC\_CHECK1);  hwndRadio = GetDlgItem(hDlg,IDC\_RADIO1);  LRESULT lCheck;  switch(uMsg)  {  case WM\_COMMAND:  cmdId = LOWORD(wParam);  switch(cmdId)  {  case IDOK://防止用户按回车退出程序  if(IDOK == MessageBox(hDlg,TEXT("Exit App?"),TEXT("Info"),MB\_OKCANCEL))  {  EndDialog(hDlg,FALSE);  }  break;  case IDC\_BUTTON1:  MessageBox(hDlg,TEXT("Button Clicked"),TEXT("Info"),MB\_OK);  break;  case IDC\_CHECK1:  MessageBox(hDlg,TEXT("CheckBox Clicked"),TEXT("Info"),MB\_OK);  break;  case IDC\_RADIO1:  MessageBox(hDlg,TEXT("RadioButton Clicked"),TEXT("Info"),MB\_OK);  break;  case IDC\_BTN\_SET:  SetWindowText(hwndBtn,"Setted...");  break;  case IDC\_BTN\_GETTEXT:  GetWindowText(hwndBtn,szCaption,128);  MessageBox(hDlg,szCaption,TEXT("Info"),MB\_OK);  break;  case IDC\_BTN\_SHOW\_HIDE:  /\*ToggleShowWin(hwndBtn);  ToggleShowWin(hwndChk);  ToggleShowWin(hwndRadio);\*/  ShowWindow(hwndBtn,!IsWindowVisible(hwndBtn));//更好的写法，不需要自己定义函数  ShowWindow(hwndChk,!IsWindowVisible(hwndChk));  ShowWindow(hwndRadio,!IsWindowVisible(hwndRadio));  break;  case IDC\_EN\_DISABLE:  //ToggleEnableCtrl(hwndBtn);  //ToggleEnableCtrl(hwndChk);  //ToggleEnableCtrl(hwndRadio);  EnableWindow(hwndBtn,!IsWindowEnabled(hwndBtn)); //更好的写法，不需要自己定义函数  EnableWindow(hwndChk,!IsWindowEnabled(hwndChk));  EnableWindow(hwndRadio,!IsWindowEnabled(hwndRadio));  break;  case IDC\_BTN\_CHECK:  SendMessage(hwndChk,BM\_SETCHECK,BST\_CHECKED,0);//这里给wParam设置BST\_CHECKED，也就是1  SendMessage(hwndRadio,BM\_SETCHECK,BST\_CHECKED,0);  break;  case IDC\_BTN\_UNCHECK:  SendMessage(hwndChk,BM\_SETCHECK,BST\_UNCHECKED,0);//这里给wParam设置BST\_UNCHECKED。也就是0  SendMessage(hwndRadio,BM\_SETCHECK,BST\_UNCHECKED,0);  break;  case IDC\_BTN\_GETSTATE:  lCheck = SendMessage(hwndChk,BM\_GETCHECK,0,0);//获取选中状态需要接收SendMessage函数的返回值  if(lCheck)  {  MessageBox(hDlg,TEXT("Checked"),TEXT("Info"),MB\_OK);  }  else  {  MessageBox(hDlg,TEXT("UnChecked"),TEXT("Info"),MB\_OK);  }  break;  }  break;  case WM\_CLOSE: //用户点击x按钮直接退出程序  EndDialog(hDlg,FALSE);  default:  ret = FALSE;  break;  }  return ret;  }  int APIENTRY WinMain(HINSTANCE hInstance,HINSTANCE prec,LPTSTR lpCmdLine,int nCmdShow)  {  DialogBox(hInstance,MAKEINTRESOURCE(IDD\_DIALOG1),NULL,Dlgproc);  return 0;  } |