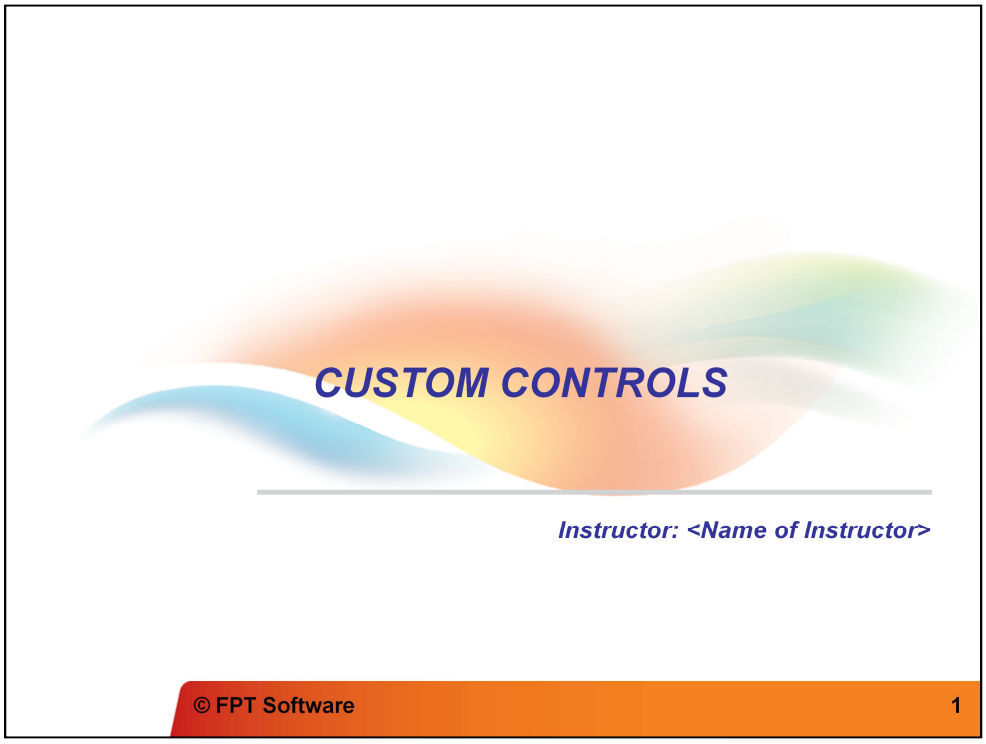
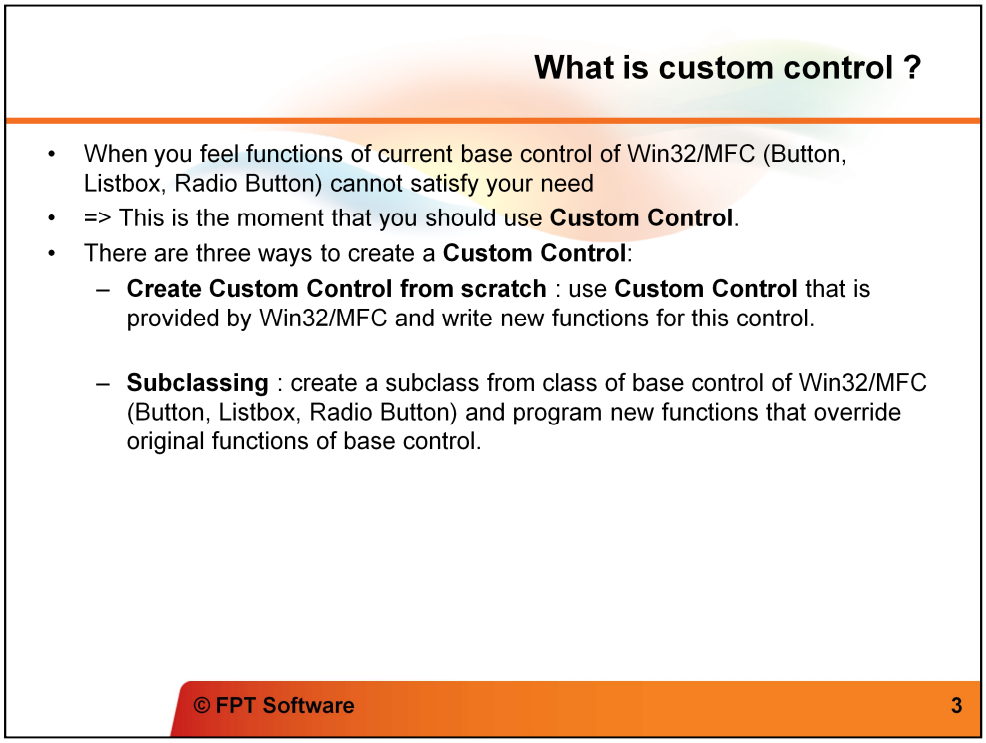


Latest updated by: HanhTT1

Agenda

- What is custom control?
- Create Custom Control in Win32 Application
- Subclassing
  - MFC Application
  - Win32 Application



Subclassing : You only use this way when you satisfy with current base control and just want an improvement or add-in for them.

A **Custom Control** can contain many useful properties and features if you have good knowledge about programming.

Create Custom Control in Win32 application

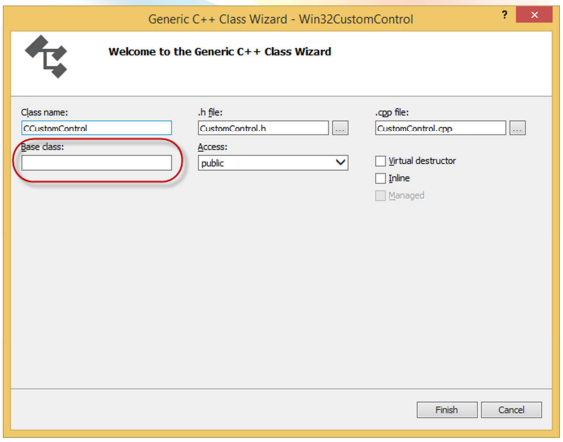
- **Custom control** is a sub item of **Dialog Editor** pattern of **Toolbox** in Visual Studio
- The main difference between **Custom Controls** and other **Normal Controls** is existence of class. **Normal controls** have class (default) and **Custom Controls** do not.
- In general, when creating a **Custom Control**, you must develop a **class** for it.
- Main steps when creating a **Custom Control:**
  - Create a custom class
  - Register custom class
  - Add code to customize control

A custom control can be drawn directly on dialog like other normal controls.

When a custom control is putted on dialog, if there are no class is attached to it or attached class is unregistered => application cannot start.

Create Custom Control from scratch
Win32 Application – Create Custom Class

- Use **Generic C++ Class Wizard** to add **Custom Class**



Create Custom Control from scratch
Win32 Application – Register Custom Class

- Define **Custom Class** information by setting value into member variable of **WNDCLASSEX** (**lpszClassName** &

```
void InitCustomControl()
{
    WNDCLASSEX wc;

    wc.cbSize         = sizeof(wc);
    wc.lpszClassName  = szClassName;
    wc.hInstance      = GetModuleHandle(0);
    wc.lpfnWndProc    = CustWndProc;
    wc.hCursor        = LoadCursor (NULL, IDC_ARROW);
    wc.hIcon          = 0;
    wc.lpszMenuName   = 0;
    wc.hbrBackground  = (HBRUSH)GetSysColorBrush(COLOR_BTNFACE);
    wc.style          = 0;
    wc.cbClsExtra     = 0;
    wc.cbWndExtra     = 0;
    wc.hIconSm        = 0;

    RegisterClassEx(&wc);
}
```

- In window procedure of custom control, handle windows message and add code to customize control.

```
LRESULT CALLBACK CustWndProc(HWND hwnd, UINT msg, WPARAM wParam, LPARAM lParam)
{
    switch(msg)
    {
        case WM_NCCREATE:
            break;
        case WM_PAINT:
            //Draw custom GUI for Custom Control
            return CustCtrl_OnPaint(hwnd);
            break;
        default:
            break;
    }

    return DefWindowProc(hwnd, msg, wParam, lParam);
}
```

Subclassing

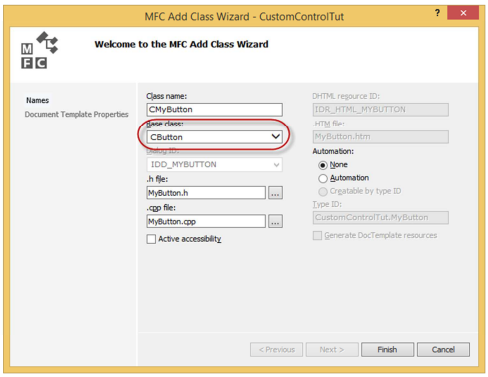
- **Subclassing** an existing control is another way to create a custom control.
- In a other hand, **subclassing** is make use of properties of original controls and change, add or modify these properties to satisfy your need instead of developing from scratch.
- Below section introduces method to create custom control by subclass in MFC application and Win32 application

If a default control does almost everything you want, but you need a few more features, you can change or add features to the original control by subclassing it.

By example, you want to add a bitmap next to button control of **MFC**. In normal way, you will create a new custom control and customize it by coding all properties of normal button control + bitmap property => this way will waste a lot of time and the result may be awry. Hence, the best solution is **subclassing**.

Subclassing
MFC Application

- **Create a custom class which** inherits **base class** of original control.
- Below example show how to create custom button from CButton of MFC



Main steps of **Subclassing** in MFC same with creating **Custom Control** from scratch, except **Create Custom Class** steps.

Subclassing
Win32 Application

- In Win32 application, first creating a window procedure method to handle event of custom control
- Use **SetWindowLong** method to register **Window Procedure** method of **Custom Class** with handle of **original control**.
- Below is sample code to register a CustomButtonProc to a control whose handle is hwnd

```
void SubclassButton(HWND hwnd)
{
    LONG oriButtonProc;

    // Remember old window procedure
    oriButtonProc = GetWindowLong(hwnd, GWL_WNDPROC);
    SetWindowLong(hwnd, GWL_USERDATA, oriButtonProc);

    // Perform the subclass
    oriButtonProc = SetWindowLong(hwnd, GWL_WNDPROC, (LONG)CustomButtonProc);
}
```


Q&A