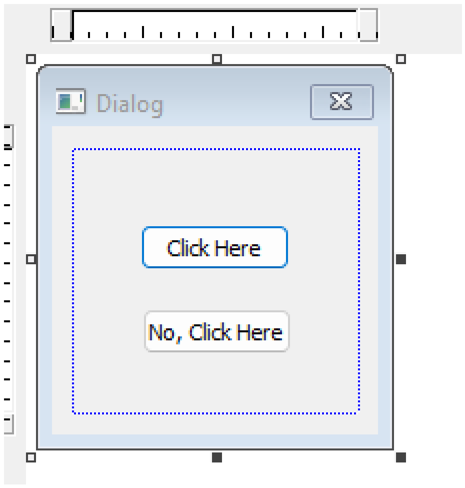
Modeless Dialogs

`DialogBox()` created modal dialogs with its on modal loop that stops any other message outside of this loop from being processed. In contrasts `CreateDialog()` behaves more like a new window that allows for message returns of the main window, this is called modeless.

First we can start by creating a dialog just like before, either by editing the `.rc` file or

```
IDD_MLDIALOG1 DIALOGEX 0, 0, 109, 95
STYLE DS_SETFONT | DS_MODALFRAME | DS_FIXEDSYS | WS_POPUP | WS_CAPTION | WS_SYSMENU
CAPTION "Dialog"
FONT 8, "MS Shell Dlg", 400, 0, 0x1
BEGIN
    DEFPUSHBUTTON   "Click Here",IDOK,29,30,50,14
    PUSHBUTTON      "No, Click Here",IDCANCEL,30,56,50,14
END
```

using the resource editor from visual studio.



For this test we'll be creating this dialog as soon as the appliation opens. So its creation will be handled on `VM_CREATE`

💡 In contrast with `DialogBox()` , `CreateDialog()` returns a windows handler.

The general process is

1. Create the dialog and assign the IDs to the buttons
2. Put the IDs on the header file and assing values to them.

   ```
   #define IDCLICK1                    1001
   #define IDCLICK2          1002
   ```

3. Create dialog itself using `CreateDialog()` , within that asign the dialog handler and anchor the Proc handler function.
4. Create the Proc handler in this case `ToolDlgProc`
5. Create event handlers within the switch under `VM_COMMAND` that aligns with the buttons IDs

   ```
   BOOL CALLBACK App::ToolDlgProc(HWND hwnd, UINT Message, WPARA
   M wParam, LPARAM lParam) {
       switch (Message)
       {
       case WM_COMMAND:
           switch (LOWORD(wParam))
           {
           case IDCLICK1:
               MessageBox(hwnd, "Hi!", "Wrong option", MB_OK | MB_ICONEXCL
   AMATION);
               break;
           case IDCLICK2:
               MessageBox(hwnd, "Hi!", "You should have pressed the other on
   e", MB_OK | MB_ICONEXCLAMATION);
               break;
           }
           break;
   ```

```
    default:
        return FALSE;
    }
    return TRUE;
}
```

I had a lot of issues with the variable declaration for the `toolbar1Handle_`

```
toolbar1Handle_ = CreateDialog(GetModuleHandle(NULL), MAKEINTRESOURCE(IDD_MLDIALOG1), hwnd, ToolDlgProc);
```

It had to comply with a different set of things

1. Be on the class definition
2. Be `inline`
3. have an out of class definition `HWND App::toolbar1Handle_ = nullptr;` on `App.cpp`

To have the window close when the [x] is clicked I just added

```
    case WM_CLOSE:
    {
        DestroyWindow(toolbar1Handle_);
        break;
    }
```

to its Proc Handler.