Password Protection

此篇文章說明如何在程式中加入密碼保護的機制，當程式開啓，使用者必須先輸入使用者帳號及密碼，若是合法使用者才能進入應用程式。

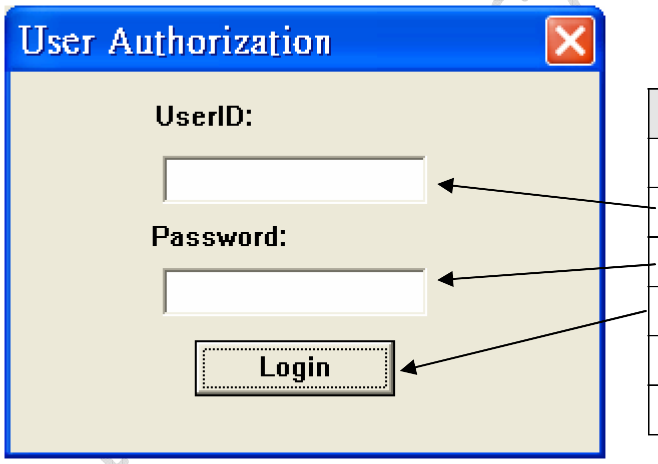
Step 1. 使用 Visual C++ 6.0 產生一個 MFC Application

1) Project name: PasswordProtection

2) Project type: MFC AppWizard(exe)

<1> What type of application would you like to create ?	Single Document
<2> What database support would you like to include ?	None
<3> What compound document support would you like to include ?	None
What other support would you like to include ?	ActiveX Controls
<4> What features would you like to include ?	3D controls
<5> What style of project would you like ?	MFC Standard
Would you like to generate source file comments ?	Yes, please
How would you like to use the MFC library ?	As a shared DLL

Step 2. 在 Project 中加入一個對話盒(Dialog Box)



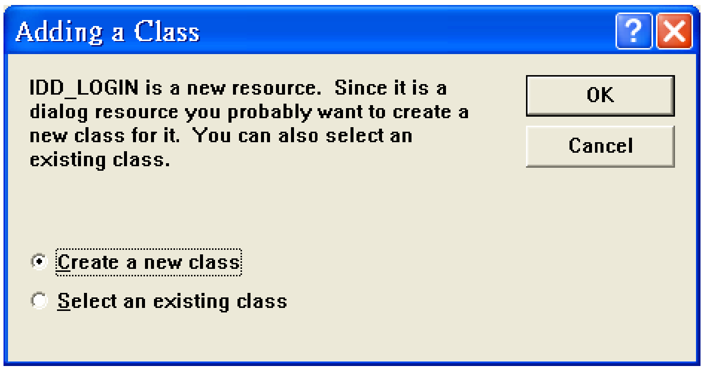
Controls	ID
Dialog Box	IDD_LOGIN
Edit Box	IDC_USERID
Edit Box	IDC_PASSWORD
Button	IDC_LOGIN
Static Text	IDC_STATIC
Static Text	IDC_STATIC

Note:

1. 不必在意 Static 控制元件的 ID 値，因爲根本不可能在程式中用到 Static 控制元件的 ID，所以重覆的 Static 控制元件 ID 値，是沒問題的!
2. 在 Passowrd 所使用的 Edit Box，記住勾選「Password」的屬性(Property)

Step 3. 利用 ClassWizard 連接對話盒與其專屬類別

1. 選按《View/ClassWizard》命令項，進入 ClassWizard，這時候出現：

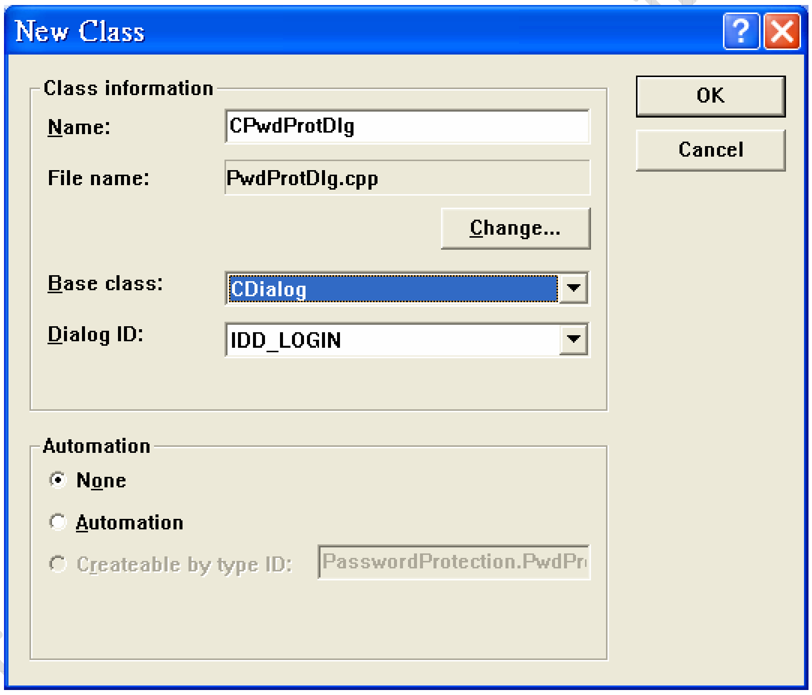


IDD_LOGIN is a new resource. Since it is a dialog resource you probably want to create a new class for it. You can also select an existing class.

[✓ Create a new class] Select an existing class

ClassWizard 知道您已在對話盒編輯器中設計一個對話盒面板，卻還未設計其對應類別，按下《Ok》，產生一個對應的新類別。

2. 在《Create New Class》對話盒中設計新類，之後選擇《Ok》



Class information
Name: CPwdProtDlg
File name: PwdProtDlg.cpp
Base class: CDialog
Dialog ID: IDD_LOGIN

Automation
[✓ None] Automation Createable by type ID: PasswordProtection.PwdPr

若按《Ok》，出現以下這個視窗：

《Unable to open the files <class>.h, <class>.cpp for class “classname”》

可參考微軟的文章

□ FIX: ClassWizard Unable to Create Files for New Class
http://support.microsoft.com/kb/196796

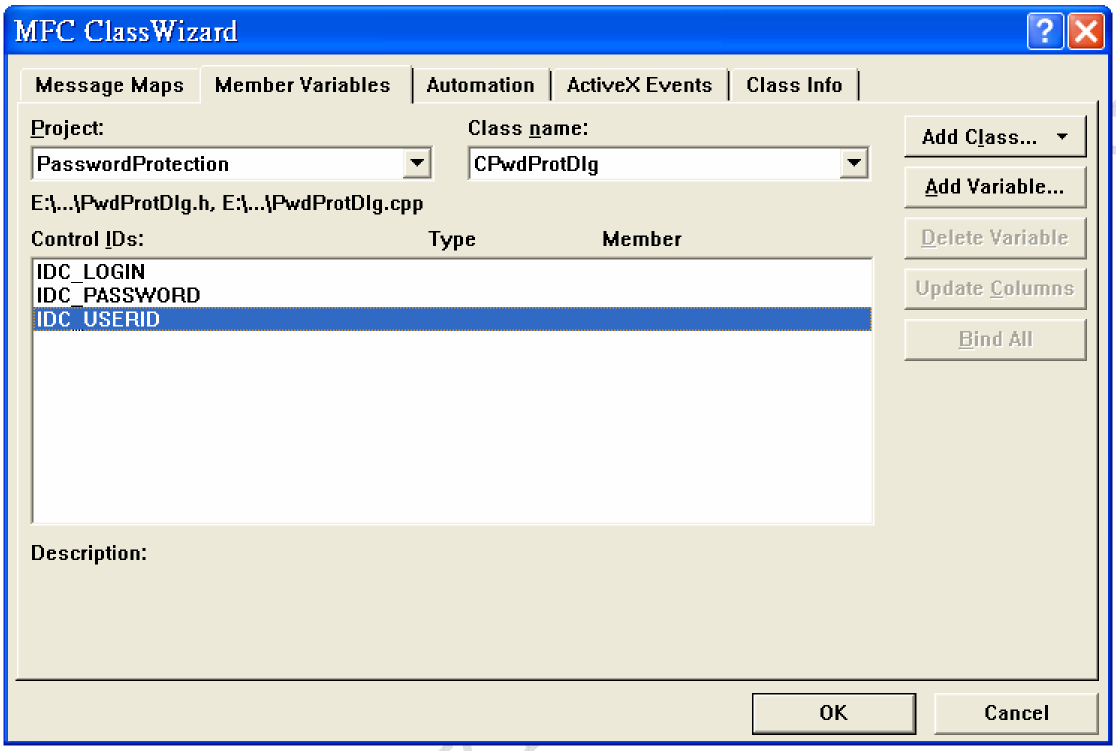
其中說到，最好的解決方法：

The best workaround is to exclude the .clw, .h, and .cpp files from being scanned by the antivirus software.

Step 4.　對話資料交換

打算以兩個成員變數映射到對話盒上的兩個 Edit Box，第一個欄位自動儲存到 m_sUserID 變數中，第二個欄位自動儲存到 m_sPassword 變數中。

1.　進入 ClassWizard，選擇《Member Variables》附頁

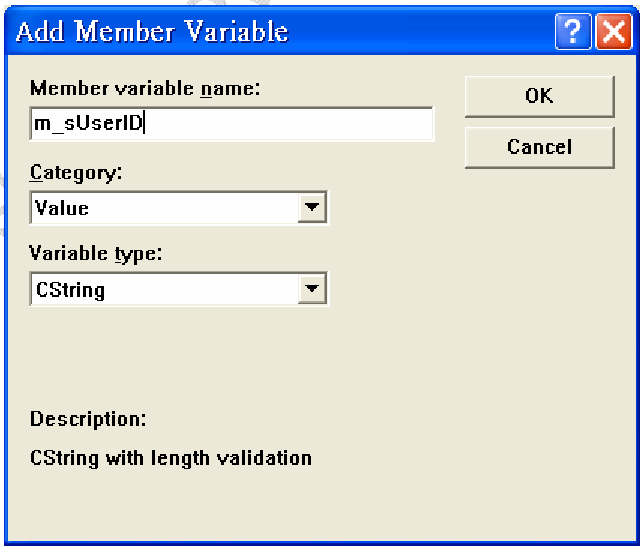


Project: PasswordProtection
Class name: CPwdProtDlg
E:\...\PwdProtDlg.h, E:\...\PwdProtDlg.cpp

Control IDs:	Type	Member
IDC_LOGIN		
IDC_PASSWORD		
IDC_USERID		

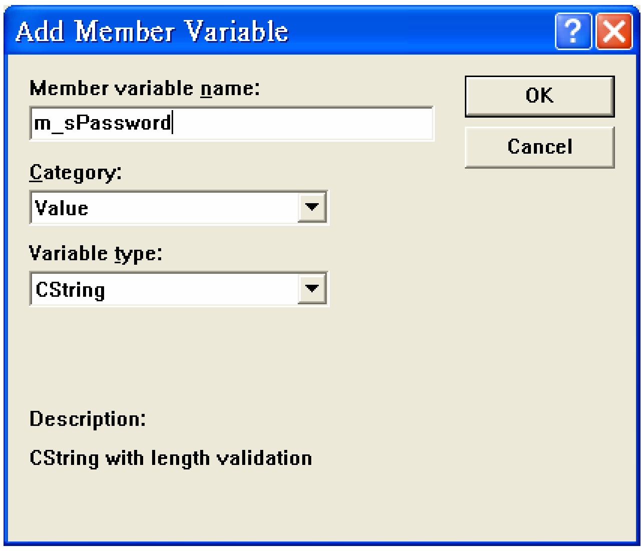
Description:

2.　選擇《IDC_USERID》，按下《Add Varaible...》按鈕：



Member variable name: m_sUserID
Category: Value
Variable type: CString
Description:
CString with length validation

3.　同樣的，選擇《IDC_PASSWORD》，按下《Addariable...》


Add Member Variable

Member variable name: m_sPassword

Category: Value

Variable type: CString

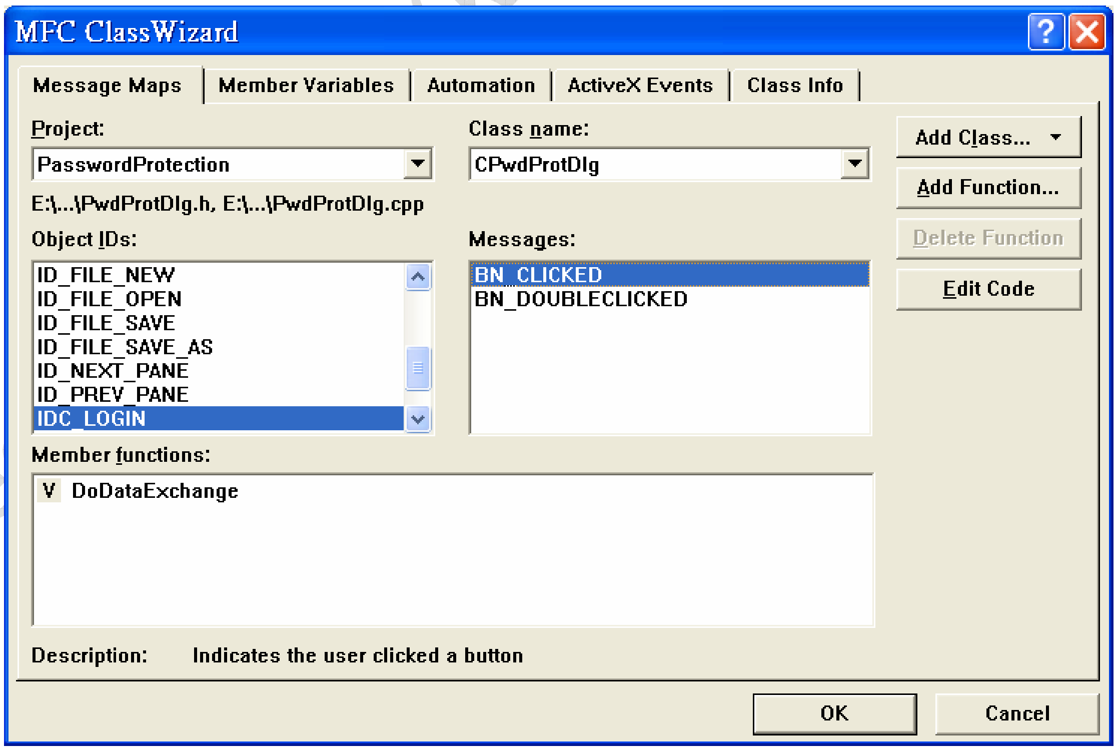
Description:

CString with length validation

Step 5. 對話盒的訊息處理函式

針對按鈕《Login》做函式對映，功能「按《Login》按鈕時，會從 user.mdb 資料庫中去檢查使用者輸入的帳號及密碼是否存在及正確。

1. 進入 ClassWizard，選擇《Message Maps》附頁，再選擇《Class Name》清單中的 CPwdProtDlg。
2. 左側的《Object IDs》清單列出對話盒中各個控制元件的 ID。選擇 IDC_LOGIN，代表《Login》的按鈕。
3. 在右側的《Message》中選擇 BN_CLICKED。


MFC ClassWizard

Project: PasswordProtection

Class name: CPwdProtDlg

E:\...\PwdProtDlg.h, E:\...\PwdProtDlg.cpp

Object IDs:

```
ID_FILE_NEW
ID_FILE_OPEN
ID_FILE_SAVE
ID_FILE_SAVE_AS
ID_NEXT_PANE
ID_PREV_PANE
IDC_LOGIN
```

Messages:

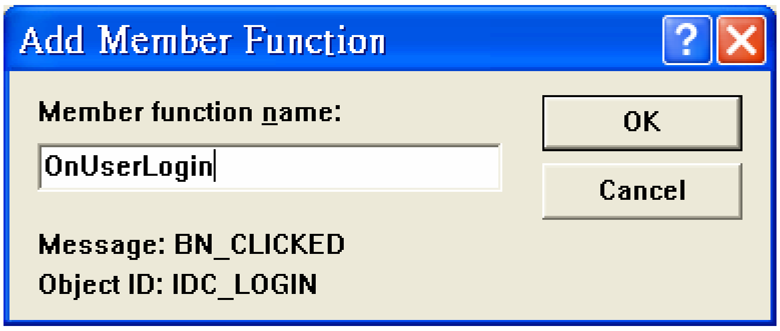
```
BN_CLICKED
BN_DOUBLECLICKED
```

Member functions:

```
V  DoDataExchange
```

Description: Indicates the user clicked a button

4. 按下《Add Function》鈕


Add Member Function

Member function name:

OnUserLogin

Message: BN_CLICKED

Object ID: IDC_LOGIN

基本的骨架已建構完成，最後的步驟就是寫程式碼。

1) In PwdProtDlg.cpp

```
void CPwdProtDlg::OnUserLogin()
{
    // TODO: Add your control notification handler code here
    CDatabase userDB;
    CString SqlString;
    CString strUserID, strUserPwd;
    CString strDriver = "MICROSOFT ACCESS DRIVER (*.mdb)";
    CString strDsn;
    CString strFile = "UserDB.mdb";

    // UserDB.mdb
    // Table name: user
    // Field:   id  |   password
    // =========================
    //       James  |  xxxxxxxxxx
    //         KIS  |  yyyyyyyyyy
    // =========================

    // Build ODBC connection string
    strDsn.Format( "ODBC;DRIVER={%s};DSN='';DBQ=%s",strDriver, strFile );

    // Open the database
    userDB.Open( NULL, false, false, strDsn );

    // Allocate the recordset
    CRecordset recset( &userDB );

    // Build the SQL statement
    SqlString = "SELECT id, password FROM user";

    // Execute the query
    recset.Open( CRecordset::forwardOnly, SqlString, CRecordset::readOnly );

    UpdateData(TRUE);

    // Loop through each record
    while( !recset.IsEOF() )
    {
        // Copy each column into a variable
        recset.GetFieldValue( "id", strUserID );
        recset.GetFieldValue( "password", strUserPwd );

        if ( m_sUserID == strUserID )
        {
            if ( m_sPassword == strUserPwd )
            {
                MessageBox( "Welcome to this system!" );
                userDB.Close();
                EndDialog( IDOK );
                return;
            }
        }

        // goto next record
        recset.MoveNext();
    }

    MessageBox( "Invalid userid or password" );
    // Close the database
    userDB.Close();
}
```

MFC function **EndDialog** is called to close the **User Authorization** dialog with the MFC-defined **IDOK** value. The value passed to **EndDialog** is the return value for MFC function **DoModal**. We pass the MFC constant **IDOK** to **EndDialog** to indicate a successful login.

2) In PasswordProtection.cpp

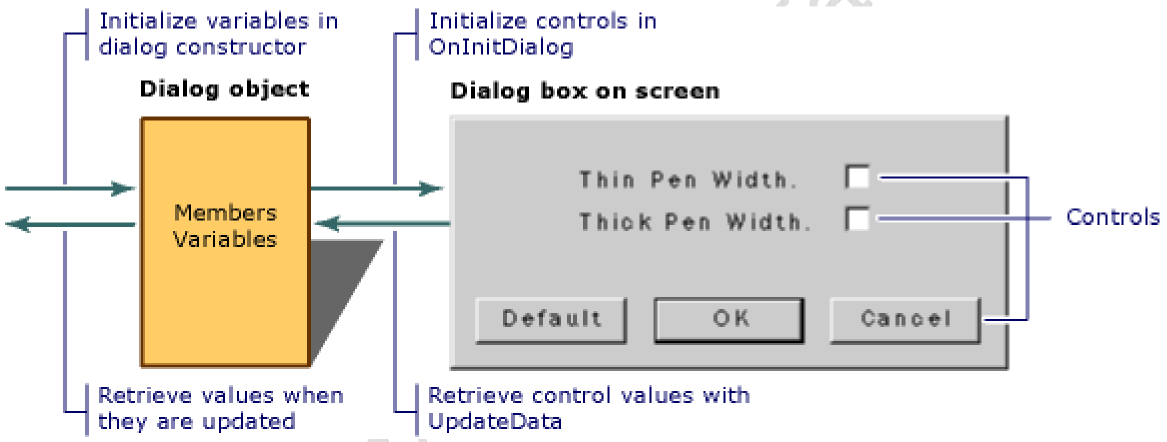
```
    #include "PwdProtDlg.h"
    . . .
    BOOL CPasswordProtectionApp::InitInstance()
    {
        AfxEnableControlContainer();

        // Standard initialization
        // If you are not using these features and wish to reduce the size
        //  of your final executable, you should remove from the following
        //  the specific initialization routines you do not need.

        #ifdef _AFXDLL
            Enable3dControls(); // Call this when using MFC in a shared DLL
        #else
            Enable3dControlsStatic();// Call this when linking to MFC statically
        #endif
        . . .

        // The one and only window has been initialized, so show and update it.
        m_pMainWnd->ShowWindow(SW_SHOW);
        m_pMainWnd->UpdateWindow();

      CPwdProtDlg loginDialg;
      if ( loginDialg.DoModal() != IDOK )
        ::PostQuitMessage(0);

        return TRUE;
    }
```

Written By Yung-Shin Liang (James Liang) @ 12/15/2006

If you use the DDX mechanism, you set the initial values of the dialog object's member variables, typically in your **OnInitDialog** handler or the dialog constructor. Immediately before the dialog is displayed, the framework's DDX mechanism transfers the values of the member variables to the controls in the dialog box, where they appear when the dialog box itself appears in response to **DoModal** or **Create**. The default implementation of **OnInitDialog** in **CDialog** calls the **UpdateData** member function of class **CWnd** to initialize the controls in the dialog box.

The same mechanism transfers values from the controls to the member variables when the user clicks the OK button (or whenever you call the **UpdateData** member function with the argument **TRUE**). The dialog data validation mechanism validates any data items for which you specified validation rules.

The following figure illustrates dialog data exchange.


Initialize variables in dialog constructor
Initialize controls in OnInitDialog
Dialog object
Dialog box on screen
Members Variables
Thin Pen Width.
Thick Pen Width.
Controls
Default
OK
Cancel
Retrieve values when they are updated
Retrieve control values with UpdateData

UpdateData works in both directions, as specified by the **BOOL** parameter passed to it. To carry out the exchange, **UpdateData** sets up a **CDataExchange** object and calls your dialog class's override of **CDialog**'s **DoDataExchange** member function. **DoDataExchange** takes an argument of type **CDataExchange**. The **CDataExchange** object passed to **UpdateData** represents the context of the exchange, defining such information as the direction of the exchange.

When you (or a Code wizard) override **DoDataExchange**, you specify a call to one DDX function per data member (control). Each DDX function knows how to exchange data in both directions based on the context supplied by the **CDataExchange** argument passed to your `DoDataExchange` by **UpdateData**.

MFC provides many DDX functions for different kinds of exchange. The following example shows a `DoDataExchange` override in which two DDX functions and one DDV function are called:

```
void CMyDialog::DoDataExchange(CDataExchange* pDX)

{ CDialog::DoDataExchange(pDX); // Call base class version DDX_Check(pDX,

IDC_MY_CHECKBOX, m_bVar); DDX_Text(pDX, IDC_MY_TEXTBOX, m_strName);

DDV_MaxChars(pDX, m_strName, 20); }
```

The DDX_ and DDV_ lines are a data map. The sample DDX and DDV functions shown are for a check-box control and an edit-box control, respectively.

If the user cancels a modal dialog box, the **OnCancel** member function terminates the dialog box and **DoModal** returns the value **IDCANCEL**. In that case, no data is exchanged between the dialog box and the dialog object.

Note:

1. 將變數(如 m_sUserID)的內容寫至對應的控制元件(Edit Box: IDC_USERID)
   UpdateData(FALSE);

2. 將控制元件(Edit Box: IDC_USERID)寫至對應的變數(如 m_sUserID)
   UpdateData(TRUE);

參考資料：

1. FIX: ClassWizard Unable to Create Files for New Class
   http://support.microsoft.com/kb/196796
2. Using the CDatabase class to read an Access databases
   http://www.codeproject.com/database/readdb.asp
3. 使用 DDX 與 DDV 操作控制項的資料
   http://debut.cis.nctu.edu.tw/~ching/Course/AdvancedC++Course/__Page/Slides/00%20Window%20and%20Message/01%20Using%20DDX%20and%20DDV.pdf
4. MFC Library Reference - Dialog Data Exchange @ MSDN
5. MFC Library Reference - Dialog Data Validation@ MSDN
6. Chapter 3 @ Getting Started with Microsoft Visual C++ 6 with an Introduction to MFC
7. Chapter 10@ 深入淺出 MFC