الفصل الحادى عشر
العمل مع المربعات الحوارية

تعاملنا فى الفصول السابقة من الكتاب مع تطبيقات تحتوى على مربع حوارى واحد فقط، إلا أنه يمكنك إنشاء المزيد من المربعات الحوارية المرتبطة بالمربع الحوارى الأساسى بإنشاء قالب وتصنيف مستقل لكل مربع حوارى جديد، ومن ثَّم إضافة أىٍ من أدوات التحكم الأخرى والتفاعل معها داخل المربع الحوارى.

بانتهاء هذا الفصل ستتعرف على:

- إنشاء المربعات الحوارية المخصصة وتصنيفات الاحتواء المصاحبة لكل منها.
- كيفية استخدم المربعات الحوارية المنبثقة (Modal) لاحتواء بيانات التطبيق.
- إنشاء واستخدام المربعات الحوارية الغير منبثقة فى الحالات الخاصة.
- استخدام المربعات الحوارية لتبادل البيانات والتحقق من صحتها.

يمكن تقسيم المربعات الحوارية إلى نوعين أساسيين، الأول يمنع التفاعل مع أى عنصر داخل التطبيق إلا بعد إغلاقه ويسمى **Modal Dialog Box**، أمـا النـوع الثـانى فـيمكن للمستخدم تنشيط عناصر التطبيق الأخرى حتى لو كان المربع مازال مفتوحـاً ويسـمى **Modeless Dialog box**. وفيما يلى سنقوم بالشرح التفصيلى لكلٍ من النوعين.

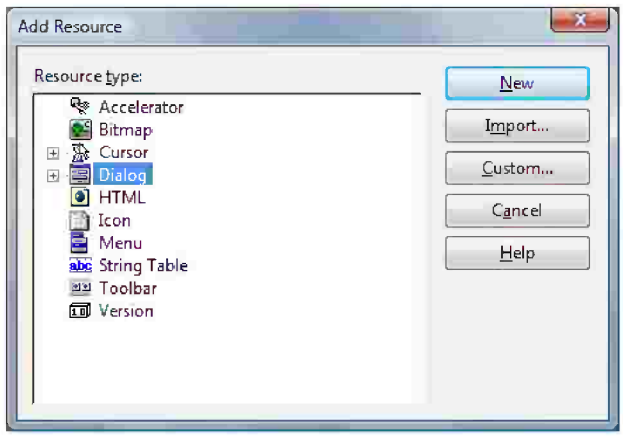
إضافة مربع حوارى جديد

قبل أن تقوم بإنشاء تصنيف المربع الحوارى، يجب أن تقوم أولاً بإنشاء قالب المربع الحوارى. تابع معنا الخطوات الآتية:

١. قم بإنشاء مشروع حوارى جديد باسم **MyDialog**.

٢. من نافذة عمل المشروع، نشط التبويـب **Resource View** إذا لم يكـن هـو التبويب النشط.

٣. انقر بزر الفأرة الأيمن أول مورد من موارد المشروع ثم اختـر **Add Resource** من القائمة الموضعية، يظهر المربع الحوارى **Add Resource** (انظر شكل ١١-١).



شكل ١١-١ إضافة مربع حوارى جديد

٤. اختر العنصر **Dialog** إذا أردت إضافة مربع حوارى عادى أو يمكنك بدلاً من ذلك نقر زر التوسيع المجاور للعنصر **Dialog** واختيار أحد المربعات الحوارية الخاصة الموجودة بالقائمة.

٥. انقر زر **New** لإضافة مربع حوارى جديد. يجب أن يظهر الآن اسم المربع الحوارى الجديد أسفل العنصر **Dialog**.

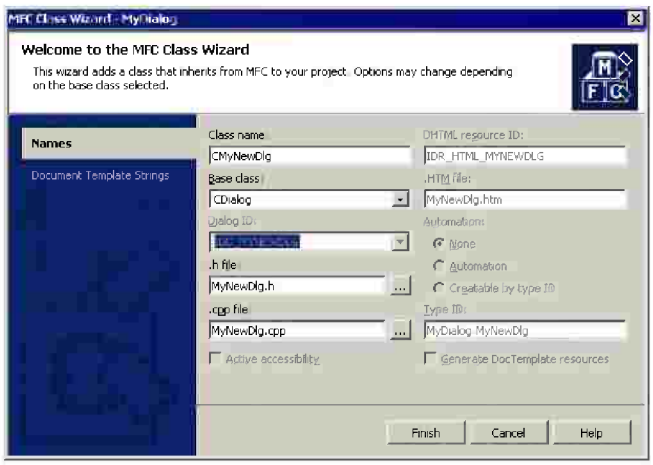
٦. انقر اسم المربع الحوارى الجديد بزر الفأرة الأيمن ثم اختر **Properties** من القائمة الموضعية، يظهر مربع الخصائص المصاحب للمربع الحوارى الجديد.

٧. قم بتغير الاسم الافتراضى للمربع الحوارى الجديد ليكون معبراً وليكن **IDD_MYNEWDLG** وذلك داخل مربع النص **ID**.

إنشاء تصنيف المربع الحوارى الجديد

بعد أن قمت بإنشاء قالب المربع الحوارى الجديد، يمكنك إضافة أى من أدوات التحكم إليه، كما يمكنك إذا أردت تغيير خصائصه الافتراضية، إلا أنك لن تتمكن من استخدامه قبل أن تقوم بربطه بتصنيف جديد كى تستطيع تحميله وإظهاره والتفاعل مع الأحداث والرسائل التى تصدر منه أو من أى أداة تحكم بداخله. يتم إنشاء هذا التصنيف من التصنيف الأساسى للمربعات الحوارية المسمى **CDialog** وذلك باستخدام معالج التصنيفات **Class Wizard** .لإنشاء تصنيف المربع الحوارى الذى أنشأناه منذ قليل، تابع معنا الخطوات الآتية:

١. تأكد من فتح قالب المربع الحوارى ثم انقر أى مكان خالى داخل القالب بزر الفأرة الأيمن واختر **Add Class** من القائمة الموضعية، تظهر نافذة معالج التصنيفات **MFC Class Wizard** الذى يمكنك استخدامه فى ربط التصنيف الجديد بالمربع الحوارى (انظر شكل ١١-٢).



شكل ١١-٢ نافذة معالج التصنيفات

٢. قم بإدخال اسم التصنيف داخل مربع النص **Class Name** على أن يبدأ بحرف **C** وليكن **CMyNewDlg**.

٣. اختر التصنيف الأساسى **CDialog** من مربع السرد والتحرير **Base Class**.

٤. انقر زر **Finish**، يقوم المعالج بإنشاء التصنيف الجديد كما يقوم أيضاً بتجهيز الملف الرئيسى للتصنيف باسم **MyNewDlg.cpp** وملف العناوين **MyNewDlg.h** وإضافتها إلى مشروعك.

وبهذا يمكنك ربط عناصر المربع الحوارى الجديد بالمتغيرات عن طريق إضافة هذه المتغيرات لتصنيف المربع الحوارى كما يمكنك إنشاء كائنات جديدة من هذا التصنيف داخل الكود.

إعطاء قيم ابتدائية لتصنيف المربع الحوارى

بعد أن قمت بإنشاء تصنيف المربع الحوارى الجديد، يمكنك مشاهدة التصنيف من خلال قائمة التصنيفات الموجودة بالتبويب **Class View**. انقر اسم التصنيف بالجزء العلوى من التبويب، تظهر دالة إنشاءه والتى تحتوى كما تعلم على نفس الاسم بالجزء السفلى. انقر دالة الإنشاء **CMyNewDlg()** نقراً مزدوجاً، يظهر كود الدالة كما يلى:

```
CMyNewDlg::CMyNewDlg(CWnd* pParent /*=NULL*/)
        : CDialog(CMyNewDlg::IDD, pParent)
{
```

```
}
```

يتم تمرير اسم المربع الحوارى CMyNewDlg::IDD للدالة حيث يتم تعريفة داخل التصنيف فى الصورة enum { IDD= IDD_MYNEWDLG}. إذا قمت بإضافة متغيرات جديدة فيما بعد، يتم وضع القيم الابتدائية لهذه المتغيرات خارج أقواس الدالة كما فى الكود التالى:

```
CMyNewDlg::CMyNewDlg(CWnd* pParent /*=NULL*/)
        : CDialog(CMyNewDlg::IDD, pParent)
        , m_strName(_T(""))
        , m_nAge(0)
{

}
```

يجب أن تكون حذراً عند تعديل الكود المنشأ بواسطة معالج التصنيفات، فعدم وجود معرف من المعرفات أو متغير من المتغيرات سيتسبب فى عدم تنفيذ التصنيف لمهامه. فإذا قمت بإضافة أى من القيم الابتدائية بنفسك، فننصح بإضافتها بعد تعليقات المعالج مباشرةً.

إظهار المربع الحوارى المنبثق (Modal Dialog Box)

بعد أن انتهيت من إنشاء قالب المربع الحوارى وتصنيفه، يأتى سؤال هام وهو كيف يتم إظهار المربع الحوارى؟ يتم ذلك عندما يقوم المستخدم باختيار أحد خيارات قائمة من القوائم أو حينما يقوم بنقر زر من الأزرار الموجودة بالنافذة الأم، وهذا ما سيظهر للمستخدم العادى. لكن ماذا سيحدث داخل كود البرنامج ؟. يتم إظهار المربع الحوارى على مرحلتين أساسيتين وذلك كما يلى:

- إنشاء كائن من كائنات التصنيف الجديد داخل ملف المربع الحوارى الرئيسى

ففى المثال الذى بين أيدينا، قم بإضافة السطر CMyNewDlg dlgMyNew; إلى ملف المربع الحوارى الرئيسى MyDialogDlg.cpp وقم أيضاً بتضمين الملف

الـــذى يحـــوى التصـــنيف header file ممـــثلاً فى الســـطر #include "MyNewDlg.h" إلى بداية الملف نفسه.

- إظهار وبدأ تشغيل المربع الحوارى

وذلك باستدعاء الدالة ()DoModal التى تتسبب فى عرض المربع الحـــوارى بمـــا يحويه من أدوات تحكم ثم انتظار أى رسالة أو حدث من أىٍ من هذه الأدوات حتى يقوم المستخدم بإغلاق المربع الحوارى. إذا حدث أى خطأ أثنـــاء إظهـــار المربـــع الحوارى، تقوم الدالة ()DoModal بإرجاع القيمة 1- أو IDABORT. أما إذا لم يحدث أخطاء، فتقوم الدالة ()DoModal بإرجاع قيمتها بمجرد استدعاء الدالـــة ()EndDialog التى تقوم بإغلاق المربع الحوارى، حيث تقوم بتمرير قيمة صحيحة إلى الدالة ()DoModal.

ففى المثال الذى بين أيدينا، يمكنك إظهار المربع الحوارى IDD_MYNEWDLG باستدعاء الدالـــة ()DoModal وذلـــك بإضـــافة الســـطر التـــالى int nRetCode =dlgMyNew.DoModal(); إلى دالة ()OnBnClickedOk الخاصة بالمربع الحوارى الرئيسى. لذلك عندما يقوم المستخدم بإغلاق المربع الحوارى، يتم تخزين القيمـــة IDOK؛ إذا قام المستخدم بنقر زر OK، أو القيمةIDCANCEL ؛ إذا قام المســـتخدم بنقـــر زر Cancel، داخل المتغير الرقمى nRetCode، ومن ثمَّ يمكنك استخدام هذا المتغير لأداء مهام معينة، كأن تقوم بتخزين البيانات التى حدث لها تغيير باستخدام عبـــارة if إذا قـــام المستخدم بنقر زر OK كما فى الكود التالى:

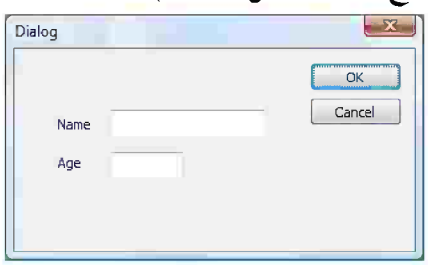
```
if (nRetCode == IDOK)
{
    // Store the dialog change
}
```

وبذلك يمكنك إغلاق المربع الحوارى باستخدام الكود فى أى وقت عن طريـــق اســـتدعاء الدالة ()EndDialog، على أن تقوم بتمرير قيمة الارتجاع المناسبة إليها.

إضافة المتغيرات لتخزين بيانات المربع الحوارى

تتطلب معظم التطبيقات أخذ نسخة من قيم أدوات التحكم الموجودة بالمربع الحوارى حتى تتيح للمستخدم تعديل هذه القيم، ومن ثمَّ تضمين القيم الجديدة إذا قام المستخدم بنقر زر **OK** أو الإبقاء على القيم القديمة إذا قام المستخدم بنقر زر **Cancel**. وهذا يعنى تمرير القيم الحالية للتطبيق لكائن المربع الحوارى قبل إظهار المربع الحوارى باستخدام الدالة **DoModal()** ثم تطبيق القيم التى حدث لها تغيير إذا كانت القيمة المرتجعة من الدالة هى **IDOK**.

فمثلاً إذا أضفنا أداتى نص إلى المربع الحوارى **IDD_MYNEWDLG**، إحداهما لتخزين الاسم والأخرى لتخزين العمر(انظر شكل ١١-٣) ثم قمنا بربط أداة نص الاسم بمتغير **m_strName** وربط أداة نص العمر بمتغير **m_nAge** من النوع **int**. (يمكنك إضافة هذه المتغيرات باستخدام المعالج كما فى شكل ١١-٤).

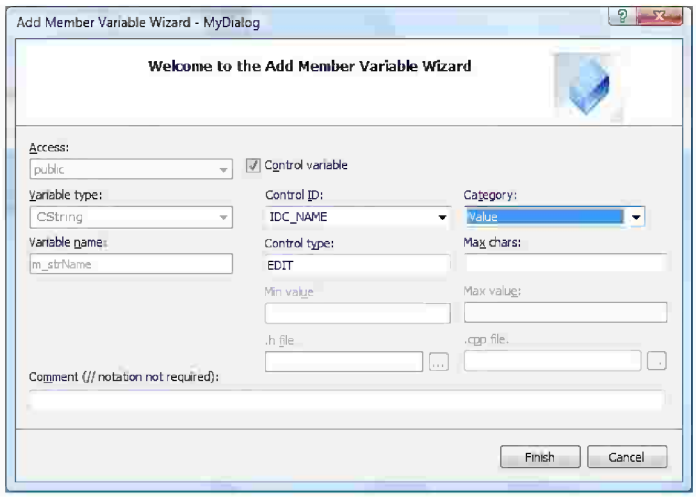


شكل ١١-٣ إضافة أداتى نص إلى المربع الحوارى

قم بتعديل كود الدالة **OnBnClickedOk()** الخاصة بالمربع الحوارى الرئيسى ليتم تخزين القيم الحالية للمربع الحوارى الجديد كما يلى:

```
void CMyDialogDlg::OnBnClickedOk()
{
   CMyNewDlg dlgMyNew(this);
   dlgMyNew.m_nAge = 27;
   dlgMyNew.m_strName = "Waleed Abd Elrazek";
   int nRetCode = dlgMyNew.DoModal();

  OnOK();
}
```



شكل ١١-٤ تعريف متغير جديد داخل المربع الحوارى

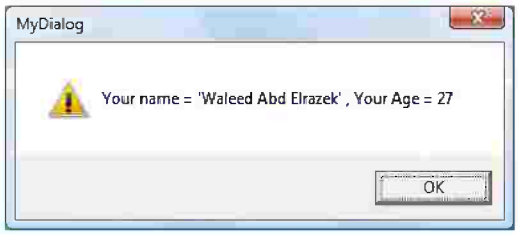
وبذلك يتم إظهار المربع الحوارى ليقوم المستخدم بالتعديل فى أدوات التحكم الموجودة فيه. فإذا قام المستخدم بإغلاق المربع الحوارى، يتم اختبار القيمة المرتجعة من الدالة **DoModal()**، فإذا كانت **IDOK** يتم تنفيذ بعض المهام. لتوضيح ذلك، قم بتعديل كود الدالة **OnOk()** كما يلى:

```
void CMyDialogDlg::OnBnClickedOk()
{
        CMyNewDlg dlgMyNew(this);
        dlgMyNew.m_nAge = 27;
        dlgMyNew.m_strName = "Waleed Abd Elrazek";
        int nRetCode = dlgMyNew.DoModal();

        if (dlgMyNew.DoModal() ==  IDOK )
         {
           CString strMsg;
           strMsg.Format("Your name = '%s' , Your Age = %d" ,
                    dlgMyNew.m_strName, dlgMyNew.m_nAge);
           AfxMessageBox(strMsg);
         }

        OnOK();
}
```

وبمجرد تشغيل التطبيق ونقر زر Ok بالمربع الحوارى الجديد، يتم تخزين القيم الجديدة فى المتغيرين dlgMyNew.m_strName وdlgMyNew.m_nAge ثم عرض رسالة تحتوى على الاسم والعمر الذى قام المستخدم بإدخاله (انظر شكل ١١-٥).



شكل ١١-٥ تظهر رسالة بالبيانات التى قام المستخدم بإدخالها

العمل مع دوال تبادل البيانات

فى البند السابق، قمنا بعرض القيم الجديدة من خلال مربع رسالة، إلا أنه فى التطبيقات الحقيقية يجب نقل هذه القيم لأماكن تخزين بيانات التطبيق. ولعلك تتساءل عن كيفية نقل هذه البيانات من المتغيرات إلى أدوات التحكم الموجودة بالمربع الحوارى أو العكس. يتم ذلك حقيقةً عن طريق الدالة DoDataExchange() التى يستخدمها المعالج لتخزين الكود المناسب لعملية نقل البيانات، أى أن هذه الدالة مسئولة عن عملية نقل البيانات فى كلا الاتجاهين.

يطلق على عملية النقل هذه تبادل البيانات Data Exchange وتبدأ بمجرد استدعاء الدالة UpdateData() التى تقوم باستقبال معامل منطقى يسمى m_bsaveAndValidate، فإذا كانت قيمة هذا المعامل FALSE؛ وهى القيمة الافتراضية، يتم تحميل أدوات التحكم من المتغيرات. أما إذا كانت قيمته TRUE، فيتم تحميل المتغيرات من أدوات التحكم. بمجرد تحميل الدالة OnInitDialog() يتم استدعاء الدالة UpdateData() بالقيمة الافتراضية لتحميل أدوات التحكم بمجرد إظهار المربع الحوارى، كما يتم استدعاء الدالة بالقيمة TRUE تلقائياً داخل الدالة

OnBnClikedOk() بمجرد نقر المستخدم للزر Ok لتحميل المتغيرات من أدوات التحكم.

وبجانب نقل البيانات، يمكنك أيضاً استخدام الدالة DoDataExchange() للتحقق من صحة البيانات التى يقوم المستخدم بإدخالها بتمرير القيمة True للدالة UpdateData(). فإذا كانت القيم المدخلة صحيحة، تقوم الدالة UpdateData() بإرجاع القيمة TRUEوإلا قامت بإرجاع القيمة FALSE إذا كانت البيانات غير صحيحة. وفيما يلى سنقوم باستعراض كلٍ من الطريقتين.

استخدام دوال تبادل البيانات DDX Functions

للتعرف على الدوال المستخدمة لتبادل البيانات، انقر الدالة DoDataExchange() من التبويب Class View نقراً مزدوجاً، تظهر محتويات الدالة داخل نافذة محرر الموارد كما يلى:

```
void CMyNewDlg::DoDataExchange(CDataExchange* pDX)
{
        CDialog::DoDataExchange(pDX);
        DDX_Text(pDX, IDC_NAME, m_strName);
        DDX_Text(pDX, IDC_AGE, m_nAge);
}
```

حيث تم تمرير المؤشر pDX إلى الدالة DoDataExchange() التى تقوم بدورها بتمرير هذا المؤشر مع اسم أداة التحكم والمتغير المرتبط بها إلى الدالة DDX_Text() التى يتم إنشاؤها من خلال بيئة التطوير لتمثيل عملية نقل البيانات.

هناك العديد من دوال DDX المستخدمة لتمثيل أنواع البيانات المختلفة ومن أشهرها الدوال الموضحة بجدول ١١-١التالى.

جدول ١١-١ دوال DDX المستخدمة للتحقق من صحة البيانات

الاسم	أداة التحكم	أنواع البيانات
DDX_Text()	مربع النص	All
DDX_Check()	مربع الاختيار	Int

الاسم	أداة التحكم	أنواع البيانات
DDX_Radio()	مجموعة خانات الخيار	Int
DDX_LBString()	مربع السرد	CString
DDX_LBStringExact()	مربع السرد	CString
DDX_CBString()	مربع السرد والتحرير	CString
DDX_CBStringExact()	مربع السرد والتحرير	CString
DDX_LBIndex	مربع السرد	Int
DDX_CBIndex	مربع السرد والتحرير	Int
DDX_Scroll	شريط التمرير	Int

تقوم بيئة التطوير بإضافة دوال DDX داخل الدلـــة DoDataExchange() نيابةً عنك. فإذا أردت إضافة دوال نقل بيانات أخرى يدوياً، قم بإضـــافتها فى نهاية الدالة DoDataExchange().

تقوم دوال DDX باختبار المؤشر pDX لمعرفة اتجاه نقل البيانات وذلك باختبـــار المـــتغير m_bsaveAndValidate كما يلى:

```
if(pDX->m_ bsaveAndValidate == TRUE)
{
    // instructions
}
```

ومن ثَمَّ يمكنك كتابة الكود اللازم لأداء مهام معينة تبعاً لمتطلبات التطبيق.

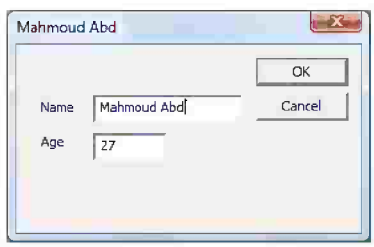
نحتاج أحياناً لاستدعاء الدالة UpdateData() داخل الكود لنقل البيانات بـــين أدوات تحكم المربع الحوارى والمتغيرات عند تمثيل دوال احتواء المربع الحوارى والتى تحتاج دائمـــاً لقيم أدوات التحكم الحالية.

فمثلاً إذا أردت إظهار اسم العميل على شريط عنوان المربع الحوارى كلما قام المســـتخدم بتغييره، قم باستخدام المعالج لإضافة دالة احتواء فى مربع نص الاسم بالمربع الحوارى الجديد

باسم OnEnChangeName() للحدث EN_CHANGE ثم قم بإضافة الكود التالى للدالة:

```
void CMyNewDlg::OnEnChangeName()
{
  UpdateData(TRUE);
  SetWindowText(m_strName);
}
```

وعلى ذلك بمجرد ضغط المستخدم لمفتاح من لوحة المفاتيح، يتم استدعاء الدالة OnEnChangeName() ومن ثمَّ تقوم الدالة UpdateData(TRUE) بتحديث المتغير m_strName بالقيمة الجديدة من خلال استدعاء الدالة DDX_Text()، وأخيراً يتم توجيه القيمة الجديدة لشريط عنوان المربع الحوارى باستخدام الدالة SetWindowText() (انظر شكل ١١-٦).



شكل ١١-٦ كتابة حروف الاسم لحظياً إلى شريط عنوان المربع الحوارى

استخدام دوال التحقق من الصحة DDV Functions

حينما تقوم بإضافة متغير إلى المربع الحوارى باستخدام المعالج، ربما تستخدم بعض المعاملات الاختيارية للتحقق من صحة البيانات التى يقوم المستخدم بإدخالها وذلك كما يلى:

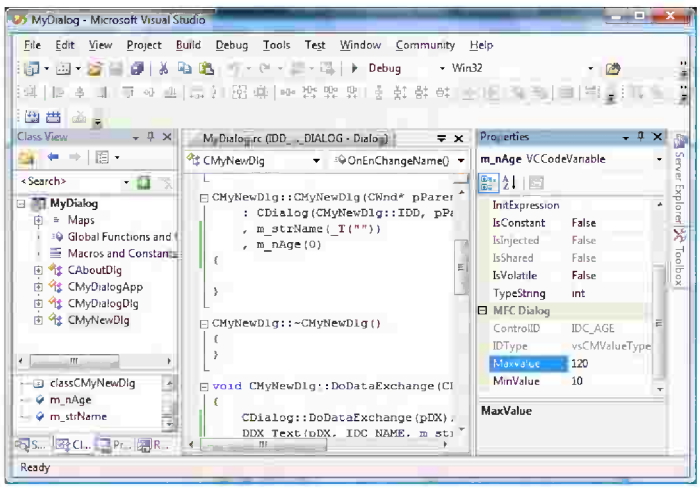
١. قم بتنشيط التبويب Class View إذا لم يكن هو التبويب النشط.

٢. انقر المتغير m_nAge بزر الفأرة الأيمن ثم اختر Properties من القائمة الموضعية لإظهار نافذة الخصائص.

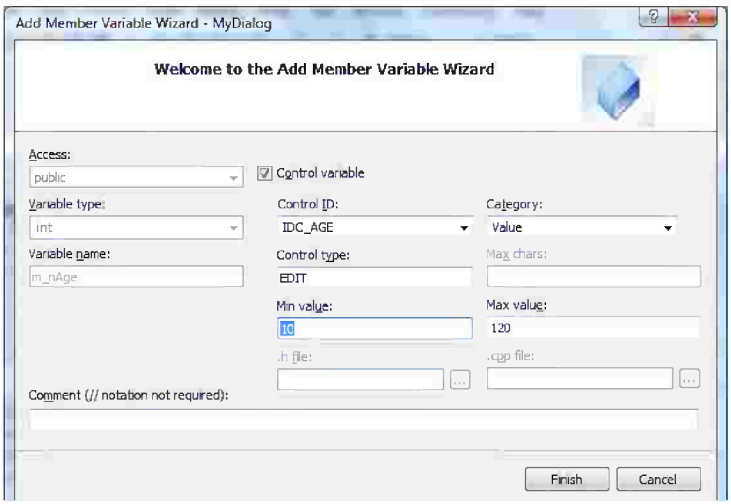
٣. قم بتعيين خيارات التحقق من الصحة داخل الخاصية MaxValue والخاصية MinValue (انظر شكل ١١-٧).

٤. يمكنك أيضاً تعيين هذه الخيارات أثناء تعريف المتغير باستخدام المعالج وذلك من خلال مربع النص **Min Value** ومربع النص **Max Value** (انظر شكل ١١-٨).

تختلف معاملات التحقق من الصحة باختلاف نوع بيانات المتغير. فمثلاً إذا كان المتغير من النوع **CString**، يكون هناك معامل واحد فقط وهو أكبر عدد من الحروف يمكن للمستخدم إدخاله. أما إذا كان المتغير من النوع **int**، فيكون هناك معاملان ، الأول يحتوى على أقل قيمة رقمية يمكن للمستخدم إدخالها والثانى يحتوى على أكبر قيمة.



شكل ١١-٧ تعيين خيارات التحقق من الصحة من مربع الخصائص

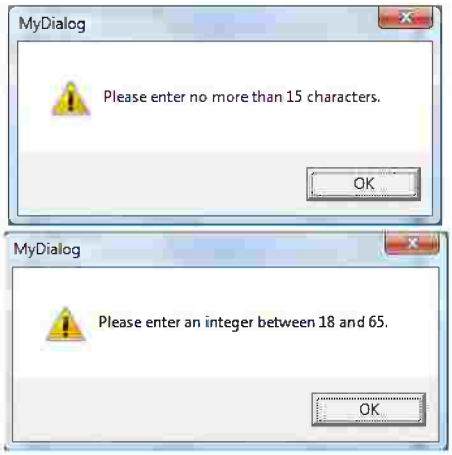


شكل ١١-٨ تعيين خيارات التحقق من الصحة أثناء تعريف المتغير

وحينما تقوم بتعيين معاملات التحقق من الصحة، يقوم المعالج نيابةً عنك بكتابة سطور DDV داخل الدالة **DoDataExchange()**. ففى المثال الذى بين أيدينا، إذا أردت ألا يزيد اسم العميل عن ١٥ حرف وألا يقل عمره عن ١٨ سنة ولا يزيد عن ٦٥، يقوم المعالج بإضافة الكود التالى:

```
DDV_MaxChars(pDX,m_strName,15);
DDV_MinMaxInt(pDX,m_nAge,18,65);
```

فإذا قام المستخدم بإدخال قيمة خارج هذا المدى، يظهر مربع رسالة يخبر المستخدم بخطأ البيانات المدخلة (انظر شكل ١١-٩).



شكل ١١-٩ تظهر رسالة الخطأ عند عدم إدخال الرقم الصحيح

يحتوى جدول ١١-٢ على دوال DDV المتاحة والتى يمكنك إضافتها يدوياً إلى نهاية الدالة DoDataExchange().

جدول ١١-٢ دوال DDV

الدالة	نوع البيانات	الاستخدام
DDV_MaxChars()	CString	التحكم فى عدد الحروف المدخلة
DDV_MinMaxByte()	BYTE	التحكم فى الرقم المدخل
DDV_MinMaxDouble()	double	التحكم فى الرقم المدخل
DDV_MinMaxDWord()	DWord	التحكم فى الرقم المدخل
DDV_MinMaxFloat()	float	التحكم فى الرقم المدخل
DDV_MinMaxInt()	int	التحكم فى الرقم المدخل
DDV_MinMaxLong()	long	التحكم فى الرقم المدخل
DDV_MinMaxUnsigned()	unsigned	التحكم فى الرقم المدخل

إنشاء دوال جديدة للتحقق من الصحة

إذا لم تتلاءم دوال **DDV** مع متطلبات تطبيقك، فبإمكانك تطوير وإنشاء دوال جديدة تتناسب مع الاختبارات التى ترغب فى إجراءها على البيانات المدخلة. تحتوى كل دالة من هذه الدوال على معاملين أساسيين، الأول عبارة عن مؤشر لكائن عضو فى التصنيف **CDataExchange** والثانى عبارة عن مرجع للمتغير المراد اختباره، بالإضافة إلى أى معاملات أخرى ترى أنك فى حاجة إليها. وعلى هذا، فمنوط بدالة التحقق من الصحة القيام بالاختبارات الآتية:

- اختبار مؤشر كائن **CDataExchange** للتحقق من أن قيمة المتغير المنطقى **m_bsaveAndValidate** هى **TRUE**. فإذا كانت هذه القيمة **FALSE**، فهذا يعنى أنه سيتم تحميل أدوات التحكم من المتغيرات، لذلك لا داعى لعملية الاختبار من الأساس ويجب إنهاء الدالة. أما إذا كانت القيمة **TRUE**، فيتم اختبار المتغير طبقاً للشروط التى قمت أنت بتحديدها.
- اختبار قيمة المتغير، فإذا كانت صحيحة ، تم إنهاء الدالة فى الوضع الطبيعى لها، وإلا يتم إظهار رسالة تحذير ثم استدعاء الدالة **Fail()** من خلال مؤشر الكائن لإخبار الدالة **UpdateData()** بعدم صحة البيانات المدخلة.

ففى المثال الذى بين أيدينا، إذا أردت إضافة شرط آخر وهو ألا يساوى عمر العميل ٣١ سنة، تكون دالة التحقق من الصحة على الشكل التالى:

```
void DDV_ValidateAge(CDataExchange* pDX,int& nCheckAge)
  {
          if(pDX->m_bSaveAndValidate && (nCheckAge < 18
          || nCheckAge > 65 || nCheckAge == 31))
     {
        AfxMessageBox("Ages must be between 18 and 65 ,
        but not 31 !);
        PDX->Fail();
     }
  }
```

حيث تم استدعاء الدالة **DDV_ValidateAge()** داخل الدالة **DoDataExchange()**

بعد تحديث المتغير بالقيمة الجديدة باستخدام دالة **DDX** وتكون الدالة على الصورة التالية:

```
DDV_ValidateAge(pDX,m_nAge);
```

العمل مع المربعات الحوارية الغير منبثقة (Modeless)

تُستخدم المربعات الحوارية الغير منبثقة **Modeless Dialog Boxes** كأشرطة أدوات غالباً، لذا يمكن للمستخدم إظهارها أو إخفاءها فى أى وقت دون التأثير على تنشيط أى من المربعات الحوارية الأخرى الموجودة بالتطبيق. وفى الواقع، لا يختلف الشكل العام لقالب المربع الحوارى المنبثق عن قالب المربع الحوارى العادى حيث يتم إنشاؤه باستخدام نفس التصنيف الأساسى للمربعات الحوارية **CDialog** إلا أن الاختلاف بين نوعى المربعات الحوارية يكمن فى عدم وجود زرى **Ok** و **Cancel** غالباً بالإضافة إلى الطريقة التى يتم بها فتح وإغلاق المربع الحوارى.

فتح وإغلاق المربع الحوارى الغير المنبثق

تختلف طريقة فتح وإغلاق المربعات الحوارية الغير منبثقة عن تلك المستخدمة مع المربعات الحوارية المنبثقة، حيث نستخدم الدالة **()Create** لإنشاء المربع الحوارى والدالة **()ShowWindow** لإظهاره على الشاشة بدلاً من استخدام الدالة **()DoModal** مع المربعات الحوارية المنبثقة.

تحتوى الدالة **()Create** على معاملين، الأول عبارة عن معرف المربع الحوارى المراد إظهاره والثانى عبارة عن مؤشر اختيارى للنافذة الأم (القيمة الافتراضية هى النافذة الأساسية للتطبيق). كما تقوم الدالة بإرجاع القيمة **TRUE** فى حالة فتح المربع الحوارى بنجاح دون أن يظهر على الشاشة، وفى هذه الحالة يجب استخدام الدالة **()ShowWindow** بتمرير المعامل **SW_SHOW**، أو القيمة **FALSE** فى حالة الفشل فى فتح المربع الحوارى. يجب أيضاً إنشاء المربع الحوارى وحذفه باستخدام كلمتى **new** و**delete** كما سنرى بعد قليل.

للتعرف على كيفية إظهار وإخفاء المربع الحوارى الغير منبثق عن قرب، تابع معنا الخطوات

الآتية:

١. قم بإنشاء مشروع حوارى جديد باسم MyModeless.

٢. قم بإضافة مربع حوارى جديد باسم IDD_MODELESS.

٣. قم بحذف زرى Ok وCancel من المربع الحوارى الجديد.

٤. قم بإنشاء تصنيف المربع الحوارى الجديد باسم CModeless.

٥. من تبويب عرض التصنيفات Class View، نشط التصنيف CModeless الذى قمت بإنشائه منذ قليل بالجزء العلوى من التبويب ثم انقر دالة إنشاء التصنيف CModeless() بالجزء السفلى نقراً مزدوجاً، تظهر الدالة داخل نافذة المحرر.

٦. قم بتعديل كود الدالة كما يلى:

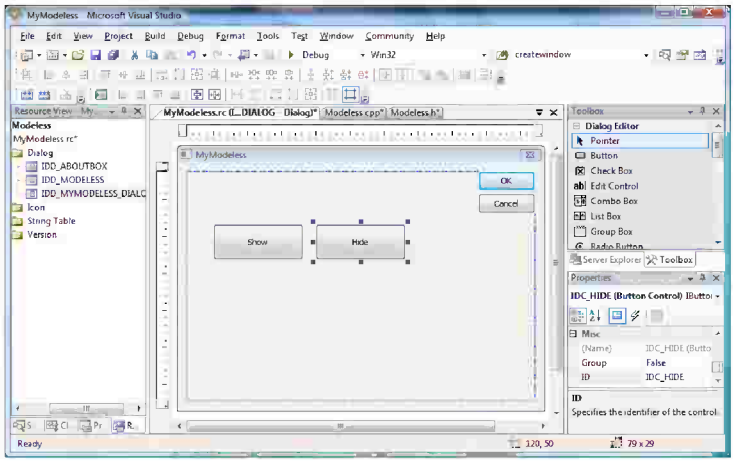
```
1.    CModeless::CModeless(CWnd* pParent /*=NULL*/)
2.    : CDialog(CModeless::IDD, pParent)
3.    {
4.      if (Create(CModeless::IDD,pParent))
5.      {
6.        ShowWindow(SW_SHOW);
7.      }
8.    }
```

وعن هذا الكود، نوضح ما يلى:

- فى السطر رقم ٤، تم استدعاء الدالة Create() لتحويل المربع الحوارى من منبثق إلى غير منبثق بتمرير معاملين، الأول عبارة عن معرف المربع الحوارى والثانى عبارة عن مؤشر للنافذة الأم.
- فى السطر رقم ٦، يتم إظهار المربع الحوارى على الشاشة فى حالة إرجاع الدالة Create() للقيمة TRUE وذلك بتمرير المعامل SW_SHOW للدالة ShowWindow().

٧. قم بإضافة زرى تحكم إلى المربع الحوارى الرئيسى، أحدهما باسم IDC_SHOW وعنوانه Show لإظهار المربع الحوارى الغير منبثق والآخر باسم IDC_HIDE

وعنوانه Hide لإخفائه (انظر شكل ١١-١٠).



شكل ١١-١٠ إضافة زرى تحكم لإظهار وإخفاء المربع الحوارى الغير منبثق

٨. قم بإنشاء دوال أحداث النقر لكلٍ من الزرين، إحداهما OnBnClickedShow() لإظهار المربع الحوارى والأخرى OnBnClickedHide() لإخفائه ثم قم بإضافة الكود التالى لكلٍ من الدالتين:

```
1.    #include "Modeless.h"
2.    CModeless* g_pDlgModeless = NULL;

3.    void CMyModelessDlg::OnBnClickedShow()
4.    {
5.      if(!g_pDlgModeless)
6.          g_pDlgModeless = new CModeless(this);
7.    }

8.    void CMyModelessDlg::OnBnClickedHide()
9.    {
10.     if(g_pDlgModeless)
11.     {
12.        delete g_pDlgModeless;
13.        g_pDlgModeless = NULL;
14.     }
15.   }
```

وعن هذا الكود، نوضح ما يلى:

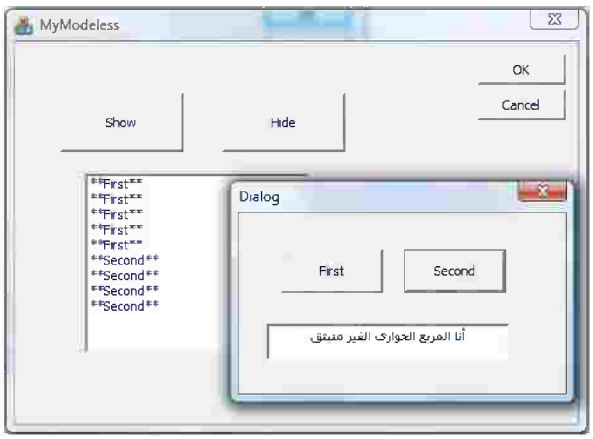
- فى السطر رقم ١ تم تضمين الملف الذى يحتوى على تعريف التصنيف CModeless وهو Modeless.h.
- فى السطر رقم ٢ تم تعريف المتغير g_pDlgModeless وهو عبارة عن مؤشر للمربع الحوارى، كما تم إعطاءه القيمة الابتدائية NULL.
- فى السطر رقم ٥ يتم اختبار قيمة المتغير g_pDlgModeless للتأكد من عدم فتح المربع الحوارى.
- فى السطر رقم ٦ تم استخدام كلمة new لإنشاء المربع الحوارى الغير منبثق بتمرير المعامل this للتعبير عن النافذة الأم.
- فى السطر رقم ١٠ يتم اختبار قيمة المتغير g_pDlgModeless للتأكد من فتح المربع الحوارى.
- فى السطر رقم ١٢ تم حذف المربع الحوارى باستخدام كلمة delete.
- فى السطر رقم ١٣ تم إعطاء المتغير g_pDlgModeless القيمة NULL للدلالة على إغلاق المربع الحوارى.

والآن قم ببناء التطبيق وتنفيذه. انقر زر Show، تلاحظ إظهار المربع الحوارى الجديد. انقر زر الإغلاق من المربع الحوارى الجديد أو زر Hide من المربع الحوارى الأساسى، تلاحظ اختفاء المربع الحوارى.

تبادل البيانات مع المربع الحوارى الغير منبثق

يمكنك تبادل البيانات بين المربع الحوارى الغير منبثق وباقى عناصر التطبيق طالما كان المربع مفتوحاً وذلك من خلال مؤشر المربع الحوارى. ويتم استخدام نفس قواعد نقل البيانات المستخدمة مع المربعات الحوارية المنبثقة باستخدام الدوال DoDataExchange() وUpdateData(). كما يمكنك استدعاء دوال التطبيق أو تغيير محتويات متغيراته من داخل المربع الحوارى الغير منبثق كنتيجة لتفاعل المستخدم مع أدوات التحكم الموجودة فيه بتمرير مؤشر للأداة المطلوبة من خلال دالة إنشاء المربع الحوارى الغير منبثق.

لتوضيح كيفية تبادل البيانات بين المربع الحوارى الغير منبثق وباقى عناصر التطبيق، سنقوم بإضافة مربع سرد للمربع الحوارى الرئيسى وسنقوم بإرسال رسائل من المربع الحوارى الغير منبثق إلى مربع السرد باستخدام زرى تحكم. كما سنقوم بإضافة مربع نص للمربع الحوارى الغير منبثق لاستقبال النص من المربع الحوارى الأساسى (انظر شكل ١١-١١). لأداء ذلك، تابع معنا الخطوات الآتية:



شكل ١١-١١تبادل البيانات بين المربع الحوارى الغير منبثق والمربع الحوارى الأساسى

١. قم بتعديل دالة إنشاء المربع الحوارى الغير منبثق كى يكون مؤشر النافذة الأم هو CMyModelessDlg بدلاً من المؤشر الحالى CWnd وذلك من خلال تعريف التصنيف CModeless داخل الملف Modeless.h كما يلى:

```
CModeless(CMyModelessDlg*pParent);//standard constructor
```

٢. قم بتضمين الملف الذى يحتوى على تعريف التصنيف CMyModelessDlg وهو MyModelessDlg.h فى بداية ملف Modeless.h كما يلى:

```
#include"MyModelessDlg.h"
```

٣. قم بتضمين الملف Resource.h فى بداية الملف MyModelessDlg.h كما يلى:

```
#include "Resource.h"
```

٤. قم بإضافة مؤشر عضو إلى التصنيف CModeless باسم m_pParent هكذا:

```
protected:
```

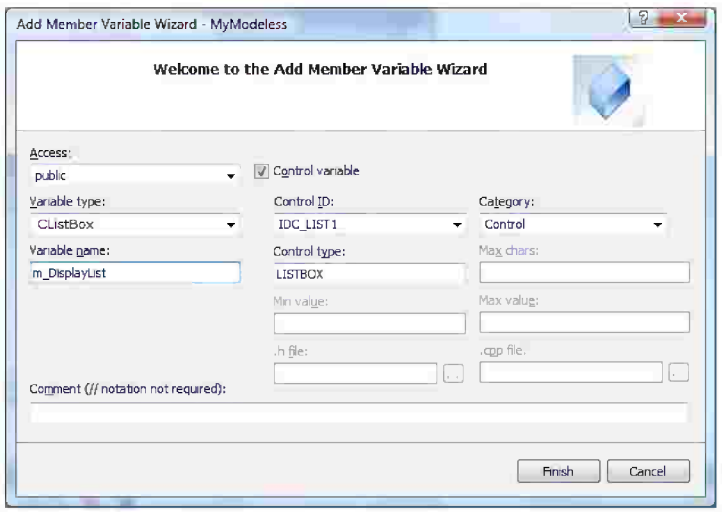
```
CMyModelessDlg* m_pParent;
```

حيث سيتم تخزين قيمة المؤشر المار من المربع الحوارى الرئيسى فى هذا المؤشر.

٥. قم بتعديل تعريف دالة إنشاء التصنيف **CModeless** لاستخدام المؤشر الجديد وذلك من خلال الملف **Modeless.cpp** كما يلى:

```
CModeless::CModeless(CMyModelessDlg* pParent /*=NULL*/)
    :CDialog(CModeless::IDD, pParent),m_pParent(pParent)
```

٦. قم بإضافة مربع سرد للمربع الحوارى الرئيسى لعرض الرسائل المرسلة من المربع الحوارى الغير منبثق. قم بربط مربع السرد بمتغير من النوع **ClistBox** باسم **m_DisplayList** (انظر شكل ١١-١٢).



شكل ١١-١٢ربط مربع السرد بمتغير باستخدام معالج التصنيفات

٧. قم بإضافة زرى تحكم للمربع الحوارى الغير منبثق أحدهما **IDC_FIRST** وعنوانه **First** والآخر **IDC_SECOND** وعنوانه **Second**.قم بإضافة دوال حدثى الزرين **OnBnClickedFirst()** و**OnBnClickedSecond()** ثم قم بتعديل أكوادها كما يلى:

```
void CModeless::OnBnClickedFirst()
{
  m_pParent->m_DisplayList.AddString("**First**");
}
```

```
void CModeless::OnBnClickedSecond()
{
  m_pParent->m_DisplayList.AddString("**Second**");
}
```

حيث يتم استدعاء الدالة **AddString()** لإضافة النصوص إلى مربع السرد الموجود بالمربع الحوارى الرئيسى والذى يشير له المؤشر **m_pParent**.

٨. للتعرف على كيفية الوصول إلى المربع الحوارى الغير منبثق من المربع الحوارى المنبثق، قم بإضافة مربع نص للمربع الحوارى الغير المنبثق ثم قم بربطه بالمتغير **m_DispMsg** من النوع **CString**.

٩. يمكنك الآن الوصول لمربع النص من أى مكان داخل التطبيق طالما كان المربع الحوارى الغير منبثق نشطاً. يمكنك مثلاً تعديل كود الدالة **OnBnClickShow()** لكتابة نص داخل مربع النص بمجرد إظهار المربع الحوارى الغير منبثق كما يلى:

```
void CMyModelessDlg::OnBnClickedShow()
{
  if(!g_pDlgModeless)
    g_pDlgModeless = new CModeless(this);
  g_pDlgModeless->m_DispMsg = CString("أنا المربع الحوارى الغير
                                        منبثق");
  g_pDlgModeless->UpdateData(FALSE);
}
```

والآن قم ببناء التطبيق وتنفيذه. انقر زر **Show**، يظهر المربع الحوارى الغير منبثق وبه الرسالة التى قمنا بتعينها. انقر زرى **First** و**Second** عدة مرات، تلاحظ ظهور الكلمات داخل مربع السرد فى المربع الحوارى الغير منبثق (راجع شكل ١١-١١).

احتواء رسالة إغلاق المربع الحوارى الغير منبثق

على الرغم من حذف زرى **Ok** و **Cancel** من المربع الحوارى الغير منبثق، يظل زر الإغلاق موجوداً، ومنه يمكنك إغلاق المربع الحوارى، وحينئذٍ للأسف لا يتم حذفه وإنما يتم إخفاؤه فقط، وبالتالى يتم تحميل الذاكرة بمربع حوارى آخر فى كل مرة تضغط فيها

على الزر Show. للتغلب على هذه المشكلة، قم بإضافة دالة الرسالة WM_CLOSE لحذف المربع الحوارى وتخصيص القيمة NULL للمؤشر. استخدم معالج التصنيفات لإنشاء الدالة ()OnClose المنبثقة من الرسالة WM_CLOSE ثم قم بتعديل كود الدالة كما يلى:

```
extern CModeless* g_pDlgModeless;
void CModeless::OnClose()
{
        CDialog::OnClose();
        delete this;
        g_pDlgModeless = NULL;
}
```

حيث تم استخدام كلمة extern لإعادة تعريف المتغير g_pDlgModeless لأننا قمنا بتعريفه من قبل.

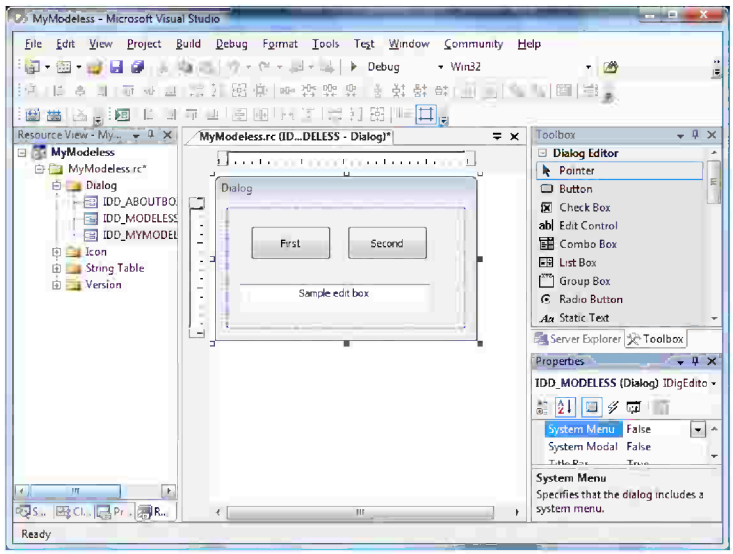
حذف خيار الإغلاق

يمكنك التغلب على المشكلة السابقة بحذف خيار زر الإغلاق من المربع الحوارى الغير منبثق من البداية وذلك كما يلى:

١. من نافذة المحرر، انقر المربع الحوارى الغير منبثق لاختياره.

٢. اضغط الاختصار Alt+Enter من لوحة المفاتيح لإظهار مربع الخصائص.

٣. قم بتخصيص القيمة False للخاصية System menu، تلاحظ اختفاء زر الإغلاق من المربع الحوارى (انظر شكل ١١-١٣).



شكل ١١-١٣حذف زر الإغلاق من المربع الحوارى