

الفصل الثاني عشر فهم المربعات الحوارية

نقدم في هذا الفصل شرحا وافيا للمربعات الحوارية المختلفة مثل مربع الرسالة، ومربع الإدخال، والمربعات الحوارية الشائعة مثل "ملف" و"طباعة" وغيره، ونشرح أيضا تصميم مربعات حوارية لأغراض خاصة تستخدم كتلك التي يستخدمها Windows أو Visual Basic.

بالانتهاء من هذا الفصل ستكتسب المعارف وتندرب على المهارات التي تجعلك قادرا على:

- إظهار معلومات للمستخدم بمربع الرسالة Message Box وكيفية الرد عليها.
- استخدام مربعات الإدخال Input Boxes.
- استخدام المربعات الحوارية الشائعة.
- تصميم المربعات الحوارية الخاصة.

المربع الحوارى Dialog Box هو نافذة صغيرة تستخدم لاستقبال المدخلات من المستخدم أو إظهار المعلومات له. ويأتي اسم الحوار من أن التطبيق والمستخدم يدخلان في حوار ليستفسر التطبيق عن أحد البيانات أو يظهر بعضها للمستخدم . يتعامل Visual Basic مع ثلاثة أنواع من المربعات الحوارية:

- مربعات الرسالة (Message Boxes) ومربعات الإدخال (Input Boxes) وهي مربعات حوارية جاهزة في Visual Basic وتستخدم عندما تريد إظهار نص بسيط أو تطلب رد المستخدم إما بإدخال سطر واحد أو باختيار أحد الأزرار. ومن أمثلتها المربعات الحوارية التي تظهر لتتأكد من جدية المستخدم الذي يطلب حذف ملف أو مجلد.
- المربعات الحوارية الشائعة (Common Dialog Boxes) وهي مبنية داخل Windows مثل مربعات اختيار الملفات، الألوان، التحكم في الطباعة ... الخ.

- مربعات حوارية خاصة، يتم تصميمها داخل نموذج جديد مع إتباع النمط المستخدم مع المربعات الحوارية.

وفي هذا الفصل سنتعرف على هذه الأنواع المختلفة من المربعات الحوارية.

مربعات إظهار المعلومات للمستخدم

تُعرض المعلومات عادة عن حالة التطبيق من خلال عناصره ونافذته الرئيسية، إلا أنه عند الحاجة إلى جذب انتباه المستخدم لمعلومة هامة أو عاجلة مثل رسالة التحذير أو الخطأ، فإن مربع الرسالة يستخدم في هذه الحالة.

مربع الرسالة عبارة عن نافذة بسيطة تعرض رسالة وزر أمر واحد أو أكثر. تستخدم هذه الأزرار ليتأكد التطبيق أن المستخدم قرأ الرسالة، ولو كان هناك قرار يجب اتخاذه بعد قراءة الرسالة فإن الأزرار تحدد اختيار المستخدم.

لإظهار مربع رسالة، تابع معنا الخطوات الآتية:

١. قم بإنشاء مشروع نوافذى جديد باسم مناسب وليكن Dialog حتى نجرى عليه التدريبات الموجودة بهذا الفصل.

٢. قم بإضافة زر أمر من مربع الأدوات إلى النموذج ثم قم بتغيير عنوانه إلى Show MessageBox.

٣. انقر الزر نقراً مزدوجاً ثم قم بكتابة الكود التالي داخل إجراء احتواء نقر الزر لإظهار مربع رسالة بمجرد نقر الزر:

MessageBox.Show("Welcome with our readers")

٤. اضغط مفتاح F5 لتشغيل التطبيق ثم انقر الزر الموجود بالنموذج، يظهر مربع الرسالة الموضح في شكل ١٢-١.

٥. انقر زر Ok لإغلاق مربع الرسالة ثم انقر زر إغلاق النافذة للعودة مرةً أخرى إلى بيئة التطوير.



شكل ١٢-١ مربع الرسالة في الوضع الافتراضي.

ولعلك تتساءل " وكيف نعرف اختيار المستخدم للأزرار؟"، في الحقيقة هناك طريقتان لاستخدام مربع الرسالة: الأولى هي عبارة `MessageBox` وهي لا ترجع قيمة، والثانية هي استخدام الوظيفة `MessageBox()` وهي ترجع قيمة تحدد الزر المختار، وسنشرح ذلك بعد قليل.

وترد على مربعات الرسالة بعض القيود رغم فائدتها في التطبيقات ومنها:

- لا تقبل مدخلات من المستخدم، فهي تظهر رسالة فقط وتكتفي بعرض وقبول خيارات محددة.
- يمكنك استخدام عدد معين من الرموز والأزرار المعرفة أساساً داخل بيئة التطوير، بينما لا يمكنك تصميم رمز أو زر أمر من عندك.
- يتوقف التطبيق حتى يرد المستخدم على الرسالة، وهذا معناه أن هذه المربعات لا يمكنك استخدامها لإظهار حالة تغيير باستمرار، لأن التطبيق ينتظر حتى يرد المستخدم ثم يستأنف عمله.

إظهار رسالة

كما عرفت فإن إظهار الرسالة بعبارة `MessageBox` هو أسهل الطرق وعند كتابة الرسالة فقط فإن الزر `Ok` فقط يظهر ولا يظهر عنوان للرسالة أو رمز يعبر عن فحواها. وحقيقةً فإن هناك بعض معاملات اختيارية تمكننا من تحديد الأزرار والرموز والعنوان الذي سيظهر على مربع الرسالة، والصيغة العامة لمربع الرسالة كما يلي:

`MessageBox.Show (prompt [, title] [, buttons])`

حيث تعبر **prompt** عن نص الرسالة بينما يعبر **title** عن عنوان الرسالة وهو اختياري أما المعامل الاختياري **buttons** فيحدد الأزرار والرمز الذي سيظهر وذلك بالاستعانة بالجدولين ١-١٢ و ٢-١٢ التاليين.

جدول ١-١٢ الثوابت المخصصة لأزرار مربعات الرسالة

| الثابت المستخدم | الأزرار | مثال للاستخدام |
|------------------|----------------------------|--|
| Ok | OK | رسالة لا تحتاج قرار من المستخدم (يظهر بها زر واحد) |
| OkCancel | OK , Cancel | تستخدم لسؤال المستخدم عن الموافقة مثل هل تريد نسخ الملفات؟ |
| AbortRetryIgnore | Abort, Retry, Ignore | تستخدم عن حدوث خطأ طارئ مثل فقدان ملف عند الإعداد وتعني (أعد القراءة، تجاهل الملف، خروج من الإعداد) |
| YesNoCancel | Yes, No, Cancel | تستخدم عند سؤال الحفظ عند إغلاق التطبيق وتعني (حفظ، إهمال، تراجع عن الخروج) |
| YesNo | Yes, No | مثل Ok, Cancel |
| RetryCancel | Retry, Cancel | يظهر بها زرین للأوامر وتسمح للمستخدم إما بإعادة العملية أو بإلغائها، والمثال على ذلك إذا لم يجد Windows القرص في مشغل القرص، فإنه يخير المستخدم بين تركيب القرص وإعادة العملية أو إلغاء العملية. |

جدول ١٢-٢ الرموز المتاحة

| نوعه | معناه | الثابت المستخدم | الرمز |
|---------------------------------------|--|--------------------------|--|
| رسالة إعلام Information Message | إعلام المستخدم بحالة التطبيق وغالباً ما تستخدم في حالة اكتمال مهمة معينة | Asterisk |  |
| رسالة حرجة Critical Message | مشكلة ضخمة حدثت، عادة ما يتم الخروج من التطبيق بعدها | Stop, Error, Hand |  |
| رسالة تحذير Warning Message | احتمال حدوث نتائج غير مرغوبة، أو خطأ يحتاج الإصلاح | Exclamation , warning |  |
| رسالة استعلام Query Message | سؤال المستخدم سؤال محدود الإجابات (نعم/لا) في الغالب | Question |  |
| رسالة إعلام Information Message | غالباً لا تحتاج اختيار من المستخدم | Information |  |

لتجربة الخيارات الموجودة بالجدولين، قم باستبدال عبارة الرسالة السابقة بالعبارة الآتية:

`MessageBox.Show("Welcome with our readers", "Welcome",
MessageBoxButtons.OK, MessageBoxIcon.Asterisk)`

حيث استخدمنا قيمة من كل جدول. قم بتشغيل التطبيق مرةً أخرى وانقر الزر الموجود

بالنموذج، تحصل على مربع الرسالة الموضح في شكل ١٢-٢ التالي.



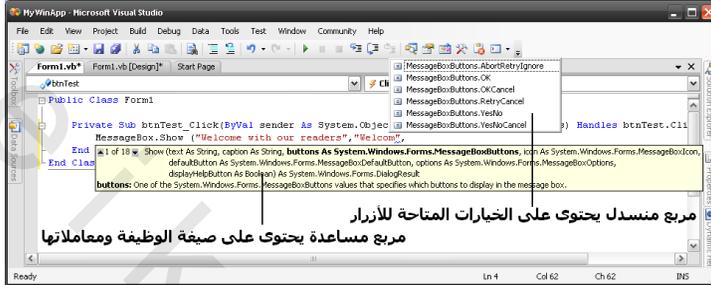
شكل ١٢-٢ مربع رسالة مكتمل الخيارات.

عليك الآن إتماماً للفائدة بتجربة الخيارات المختلفة ورؤية النتائج.

ومن تسهيلات التعامل مع نافذة البرمجة كما لاحظت أنك عند كتابتك لعبارة

`MessageBox` في نافذة الكود سيظهر لك `Visual Basic` مربع به صيغة الوظيفة

(**MessageBox.show()**) كي يسهل عليك تذكر كل من صيغة الأمر والمعاملات، فضلاً عن ذلك عند البدء في كتابة الثوابت الخاصة بالخيارات، سيظهر لك مربع منسدل به الثوابت المتاحة في هذا المكان وتسمى هذه الخاصية بخاصية الإكمال التلقائي **Auto Completion** (انظر شكل ١٢-٣).



شكل ١٢-٣ نافذة الكود ويظهر بها المربعات التي يظهرها الإكمال التلقائي.

إرجاع قيمة من الوظيفة (**MessageBox.show()**)

تستخدم الوظيفة (**MessageBox.show()**) كما أوضحنا لإعلام المستخدم بحدوث مشكلة أو حثه على اتخاذ قرار، ومع ذلك إذا أردنا معرفة أي الأزرار اختارها المستخدم من مربع الرسالة كي نحدد الخطوة التي سيتخذها التطبيق بعد ذلك، لابد من استخدام القيمة الناتجة من الوظيفة (**MessageBox.show()**). كمثال سريع قم بإضافة زر جديد إلى النموذج بعنوان **Returned Button** مع تغيير الاسم إلى **btnReturn**. انقر الزر الجديد نقرأ مزدوجاً ثم قم بإدخال الكود التالي إلى الحدث **btnReturn_Click** الخاص بنقر الزر الجديد:

```
Dim Choice As Integer
Choice = MessageBox.Show("Please choose Ok or Cancel",
"Give your choice", MessageBoxButtons.OKCancel)
If Choice = DialogResult.OK Then
    Me.Text = "You've chosen Ok"
Else
    Me.Text = "You've chosen Cancel"
End If
```

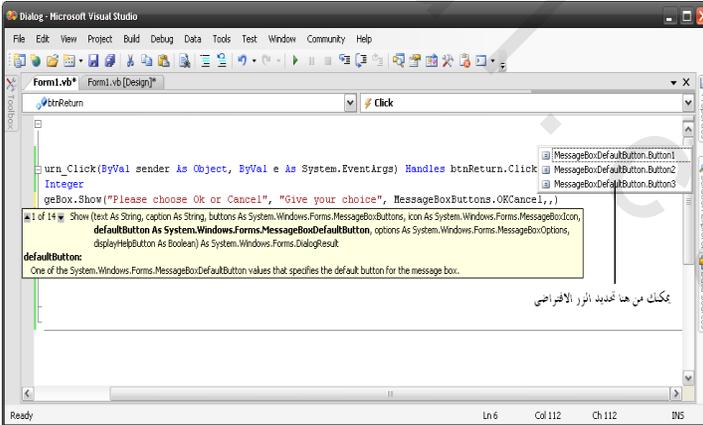
حيث يتم تخصيص ناتج مربع الرسالة إلى المتغير الصحيح **Choice** ومن ثم يتم اختبار النتيجة من خلال الخاصية **DialogResult**.

والآن قم بتشغيل التطبيق ثم انقر زر الأمر الجديد، يظهر مربع الرسالة. انقر أحد الزرين **Ok** أو **Cancel** وانظر إلى اختيارك ظاهراً في عنوان النموذج. لقد تم ذلك بمقارنة القيمة المرجعة من الوظيفة بأحد الثوابت الممثلة للأزرار في **Visual Basic** وهو **OK** ويمثل زر الموافقة، وهناك ثابت لكل زر من الأزرار التي تم ذكرها سابقاً يعرضها الجدول التالي:

جدول ١٢-٣ الثوابت المحددة للزر المختار من المستخدم في مربع الرسالة

| الثابت المستخدم | القيمة | الزر |
|----------------------------|----------|---------------|
| DialogResult.OK | 1 | OK |
| DialogResult.Cancel | 2 | Cancel |
| DialogResult.Abort | 3 | Abort |
| DialogResult.Retry | 4 | Retry |
| DialogResult.Ignore | 5 | Ignore |
| DialogResult.Yes | 6 | Yes |
| DialogResult.No | 7 | No |

لم ينته الكلام بعد عن مربع الرسالة، فهناك اختيار يحدد الزر الافتراضي (أي الذي ينوب **Enter** عنه). يمكنك من خلال الخيار قبل الأخير بمعاملات الوظيفة **MessageBox.show()** تعيين الزر الافتراضي من بين الأزرار الموجودة بمربع الرسالة والتي لا تتعدى دائماً الأزرار الثلاثة (انظر شكل ١٢-٤).



شكل ١٢-٤ تعيين الزر الافتراضي داخل مربع الرسالة.

يوضح جدول ١٢-٤ التالي خيارات الزر الافتراضي.

جدول ١٢-٤ قيم تحديد الزر الافتراضي

| الزر الافتراضي | الثابت المستخدم |
|----------------|---------------------------------|
| الأول | MessageBoxDefaultButton.Button1 |
| الثاني | MessageBoxDefaultButton.Button2 |
| الثالث | MessageBoxDefaultButton.Button3 |

المثال الأخير يجمع ذلك، لإظهار مربع رسالة مثل الموجود في شكل ١٢-٥ به رسالة تحذيرية، وبه الأزرار Yes/No ويكون الزر الثاني No هو الافتراضي، استخدم العبارة التالية:

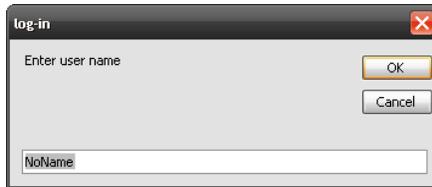
**MessageBox.Show("Are you sure?", "Warning",
MessageBoxButtons.YesNo ,MessageBoxIcon.Warning
,MessageBoxDefaultButton.Button2)**



شكل ١٢-٥ مربع رسالة به رسالة تحذيرية، والأزرار "Yes/No" والزر الثاني هو الافتراضي.

الموصول على معلومات من المستخدم

يسمح مربع الرسالة كما رأينا للمستخدم بالاختيار بين عدد من الردود على الرسالة، إلا أنه عند الحاجة إلى إدخال نص (كلمة أو رقم) كرد من المستخدم، فإن مربعات الرسالة لا تصلح لذلك. في هذه الحالة نستخدم مربعات الإدخال Input Boxes وهي تحتوي على رسالة ومربع نص للإدخال بالإضافة إلى زرین أحدهما موافق OK والآخر للإلغاء Cancel. يحتوى شكل ١٢-٦ على أحد مربعات الإدخال.



شكل ١٢-٦ مربع الإدخال InputBox.

إعداد مربع الإدخال

تحديد الرسالة والنص الافتراضي لمربع الإدخال يشبه تماماً مربع الرسالة إلا أن مربع الإدخال يعيد قيمة، ويجب معرفة هذه القيمة التي أدخلها المستخدم في المربع وذلك عن طريق تخصيص متغير لاستقبال المعلومات التي ترجع من مربع الإدخال، ثم إظهارها بعد ذلك داخل رسالة مربع الإدخال. ونوضح فيما يلي مثال لاستخدام وظيفة مربع الإدخال. للحصول على مربع الإدخال الموجود في شكل ١٢-٦ السابق، قم بإضافة زر جديد إلى النموذج بعنوان **Sample InputBox** وغير اسمه إلى **btnInputBox** ثم انقره نقرأ مزدوجاً وقم بإدخال الكود التالي داخل إجراء الحدث **btnInputBox_Click**:

```
Dim stInput As String
```

```
stInput = InputBox("Enter user name", "log-in", "NoName")
```

المعاملات الثلاثة لمربع الإدخال بالترتيب هي نص الرسالة، وعنوان المربع، والنص الافتراضي. والاثنتان الأخيرتان اختياريان أي يمكن تجاهلهما، وفي هذه الحالة يستخدم المربع اسم المشروع كعنوان للمربع والنص الافتراضي يكون خالياً.

القيم الراجعة من الوظيفة **InputBox()**

في مربع النص الموجود بمربع الإدخال يمكن للمستخدم إدخال ما لا يزيد عن ٢٥٤ حرفاً أو رمزاً، بعد الإدخال يضغط المستخدم زر موافق "OK" أو إلغاء "Cancel"، في حالة الضغط على زر OK تقوم الوظيفة بإرجاع النص المدخل، وفي حالة الضغط على زر الإلغاء، تقوم الوظيفة بإرجاع نصاً خالياً (نص طوله صفر ولا يحتوي على حروف). فكلية "Computer" على سبيل المثال تحتوي على ٨ حروف بينما لا تحتوي كلمة " " على أية حروف وهو النص الخاوي الذي نقصده. يمكنك استخدام الوظيفة **Len** لإرجاع طول النص بالحروف كما يلي:

```
Dim i As Integer
```

```
i = Len("Computer")
```

وهذه الوظيفة ذات فائدة كبيرة عند التعامل مع النصوص. ومن الدوال التي قد تستخدمها أيضاً عند التعامل مع مربع الإدخال، الوظيفة **Val()**. وهي تعيد القيمة العددية لسلسلة نصية **String**. فمثلاً الوظيفة:

```
Val("123")
```

تعطي القيمة العددية ١٢٣، والفارق أن القيمة النصية لا يمكن إجراء العمليات الحسابية

عليها، بينما القيمة العددية يمكن إجراء ذلك عليها. من الحالات التي تحتاج فيها لاستخدام الوظيفة **Val()** عندما يحتاج برنامجك لمقارنة الرقم الذي يدخله المستخدم مع رقم آخر في حالة استخدام جملة **IF**. إذا أخطأ المستخدم وأدخل حرفاً مكان الرقم فإن التطبيق يعطي رسالة خطأ.

الوظيفة **Val()** ترجع القيمة العددية لسلسلة نصية إن كانت مكونة من أرقام فقط، أما إذا احتوت على حروف ولكن تبدأ بأرقام فإنها ترجع القيمة العددية للأرقام، فإن لم تبدأ برقم فإنها ترجع صفراً سواء احتوت السلسلة النصية بعد ذلك على أرقام أم لا. المثال التالي توضيح أكثر لفكرة كل من الوظيفة **Val()** والوظيفة **Len()**. قم بإدخال الكود التالي في إجراء احتواء حدث نقر الزر **btnInputBox**:

```
Dim stInputVal As String
stInputVal = InputBox(" Enter Employee Age")
If Len(stInputVal) = 0 Then
    MessageBox.Show("No age was entered")
Else
    If Val(stInputVal) = 0 Then
        MessageBox.Show("correct the Age")
    Else
        MessageBox.Show("Continue")
    End If
End If
```

ومعنى هذه التعليمات إذا لم يدخل المستخدم أي بيانات في مربع الإدخال فسيحصل على رسالة **No age was entered** ، وإذا أدخل عبارة لا تبدأ برقم فسيحصل على رسالة **Correct the Age** وإذا أدخل عبارة رقمية أو تبدأ برقم فسيحصل على رسالة **Continue**.

سنتعرف في الباب التالي بالتفصيل على البرمجة وعلى كيفية كتابة تعليمات الكود.



استخدام المربعات الحوارية الشائعة

نشرح فيما يلي كيفية استخدام المربعات الحوارية الشائعة والتي يعرفها كافة مستخدمي **Windows**، ومن أمثلتها:

- مربع حفظ وفتح الملفات **Save/Open**

• مربع الخطوط Font

• مربع اللون Color

• مربع اختيار نوع وإعدادات الطباعة

و نظرا لشبوع هذه المربعات الحوارية، فلن نطيل في شرح مكوناتها، إنما سنشرح كيف يمكنك إظهارها في برنامجك والحصول من خلالها على اختيارات المستخدم.

المربعات الحوارية للملفات File Dialog Boxes

من أشهر المربعات الحوارية مربع "فتح Open" ويستخدم لفتح الملفات، ومربع "حفظ باسم Save As" وهو يستخدم لتحديد اسم الملف للحفظ. وهما متماثلان تقريبا، يوضح شكل ١٢-٧ مربع Open والخانات والمربعات التي يشتمل عليها.



شكل ١٢-٧ مربع "فتح Open" الذى يشبهه مربع "حفظ باسم Save As" إلى حد كبير.

الخصائص المشتركة

توجد أداتين للتحكم فى مربعات الملفات، أحدهما الأداة OpenFileDialog وقبل أن نتعرف على طريقة استخدام كل منهما، سنقوم أولاً بالتعرف على الخصائص المشتركة للأداتين وذلك من خلال الجدول ١٢-٥ التالى.

جدول ١٢-٥ الخصائص المشتركة لأداتي التعامل مع الملفات

| الخاصية | الاستخدام |
|------------------------|--|
| AddExtension | تستخدم لإضافة الامتداد تلقائياً إلى الاسم الذى يقوم المستخدم بتحديدده للملف إذا كانت قيمتها True . وهذا الامتداد يعتمد حقيقةً على عدد من العوامل. فإذا كانت قيمة الخاصية CheckFileExists هي False ، يتم استخدام أول امتداد معرف داخل خاصية الاختزال Filter . أما إذا كانت قيمتها True فيتم إضافة أول امتداد داخل خاصية Filter يتفق مع ملف موجود بالفعل. وفي أي من الحالتين إذا كانت قيمة الخاصية Filter خالية، يتم استخدام قيمة الخاصية DefaultExt . |
| CheckFileExists | إذا كانت قيمتها True وقام المستخدم بكتابة اسم غير موجود يتم إظهار تحذير مع بقاء المربع الحوارى نشطاً. أما إذا كانت قيمتها False فمن الممكن أن يقوم المستخدم بإدخال اسم غير موجود من قبل. |
| CheckPathExists | إذا كانت قيمتها True ، يتم التأكد من المسار الذى قام المستخدم بإدخاله فإذا كان خطأً يتم إظهار رسالة تحذير مع بقاء المربع الحوارى نشطاً. أما إذا كانت قيمتها False فمن الممكن أن يقوم المستخدم بإدخال اسم ملف غير موجود |
| DefaultExt | تحدد الامتداد الافتراضى الذى يظهر داخل مربع نوع الملفات File Types . وهذا هو الامتداد الذى يتم إضافته إلى اسم الملف إذا لم يقم المستخدم صراحةً بتحديد امتداد الملف |

| الخاصية | الاستخدام |
|------------------|---|
| DereferenceLinks | إذا قام المستخدم بتعيين ارتباط داخل المربع الحوارى، فإن هذه الخاصية تحدد ما إذا كان المربع سيقوم بإرجاع اسم ملف الارتباط ومكان هدف الارتباط (القيمة True أم سيقوم بإرجاع ملف الارتباط وكان الارتباط نفسه. اسم الملف فقط. فإذا كانت قيمتها True يتم إرجاع اسم الملف ومكانه |
| FileName | تعيين اسم الملف الذى سيظهر بمجرد فتح المربع الحوارى والمسار الكامل أو آخر ملف قام المستخدم باختياره |
| Filter | سلسلة بيانات تحتوى على امتدادات الملفات التى يمكن فتحها أو حفظها ووصف كل منها |
| FilterIndex | يتم من خلالها تحديد الامتداد الذى يظهر أولاً داخل مربع نوع الملفات |
| InitialDirectory | تحديد المجلد الذى يتم إظهاره بمجرد فتح المربع الحوارى |
| RestoreDirectory | حينما تكون قيمة هذه الخاصية True، يتم استعادة الدليل إلى حالته التى كان عليها قبل فتح المربع الحوارى. والقيمة الافتراضية لهذه الخاصية هي False |
| ShowHelp | حينما تكون قيمة هذه الخاصية True يتم إظهار زر مساعدة بالمربع الحوارى ومن ثمّ يمكنك كتابة كود المساعدة داخل الحدث HelpRequested |
| Title | تستخدم لتحديد النص الذى يظهر فى شريط عنوان المربع الحوارى وإلا يتم استخدام العناوين الافتراضية مثل Open أو Save As |

استخدام المربع الحوارى *Open*

للتعرف على طريقة إظهار المربع الحوارى **Open** داخل تطبيقك وكيفية استخدامه، تابع معنا الخطوات الآتية:

١. من مربع أدوات التحكم، انقر الأداة **OpenFileDialog**  نقرأ مزدوجاً لإضافتها إلى النموذج ولاحظ وضع الأداة أسفل النموذج وليس على النموذج نفسه كما فى حالة أداة القائمة المنسدلة أو أداة القائمة العادية حيث يطلق على هذه المنطقة **Tray** وتحتوى دائماً على الأدوات التى لن تظهر للمستخدم، ويتم تخصيص الأداة بالاسم **OpenFileDialog1**.
٢. قم بإضافة زر جديد إلى النموذج بعنوان **Open File Dialog** مع تغيير اسمه إلى اسم معبر وليكن **btnOpenFile**.
٣. قم بتعيين أى خصائص أخرى تراها مناسبة طبقاً للجدول ١٢-٥ السابق.
٤. انقر الزر الجديد نقرأ مزدوجاً ثم قم بإدخال الكود التالى داخل إجراء احتواء الحدث **btnOpenFile_Click**:

```
Dim sMsg As String
OpenFileDialog1.ShowDialog()
If OpenFileDialog1.FileName > " " Then
    sMsg = "The filename you chose was " &
        OpenFileDialog1.FileName
Else
    sMsg = "You did not choose a file."
End If
MessageBox.Show(sMsg)
```

٥. حيث قمنا باستدعاء الوظيفة **ShowDialog()** لعرض المربع الحوارى.
٦. قم بتشغيل التطبيق ثم انقر الزر الجديد، يتم فتح المربع الحوارى الشهير **.Open**.
٧. إذا قمت الآن باختيار أحد الملفات ثم ضغط زر **Open**، يتم إظهار رسالة تحتوى على اسم الملف ومساره الكامل. أما إذا لم تقم باختيار أى ملفات

وضغطت على الزر **Cancel** فيتم إظهار رسالة تخبرك بعدم اختيار أى ملف.

استخدام المربع الحوارى **Save As**

يمكنك إظهار المربع الحوارى **Save As** داخل تطبيقك واستخدامه بنفس طريقة استخدام المربع الحوارى **Open**. تابع معنا الخطوات الآتية:

١. من مربع أدوات التحكم، انقر الأداة **SaveFileDialog**  نقرأ مزدوجاً لإضافتها إلى النموذج حيث يتم وضع الأداة أسفل النموذج أيضاً مع تخصيصها بالاسم **SaveFileDialog1**.

٢. قم بإضافة زر جديد إلى النموذج بعنوان **Save File Dialog** مع تغيير اسمه إلى اسم معبر وليكن **btnSaveFile**.

٣. قم بتعيين أى خصائص أخرى تراها مناسبة طبقاً للجدول ١٢-٥ السابق.

٤. انقر الزر الجديد نقرأ مزدوجاً ثم قم بإدخال الكود التالى داخل إجراء احتواء الحدث **:btnSaveFile_Click**

Dim sMsg As String

SaveFileDialog1.ShowDialog()

If SaveFileDialog1.FileName > " " Then

sMsg = "The filename you chose was " &

SaveFileDialog1.FileName

Else

sMsg = "You did not choose a file."

End If

MessageBox.Show(sMsg)

حيث قمنا باستدعاء الوظيفة **ShowDialog()** أيضاً لعرض المربع الحوارى.

٥. قم بتشغيل التطبيق ثم انقر الزر الجديد، يتم فتح المربع الحوارى الشهير **Save As**.

٦. إذا قمت الآن باختيار أحد الملفات ثم ضغط زر **Save**، يتم إظهار رسالة تحتوى على اسم الملف ومساره الكامل. أما إذا لم تقم باختيار أى ملفات وضغطت على الزر **Cancel** فيتم إظهار رسالة تخبرك بعدم اختيار أى ملف.

يمكنك إجراء عمليات المعالجة المختلفة على القيمة المرجعة من مربع **Open** أو مربع **Save As** عن طريق الخاصية **FileName** التي تحتوي على اسم الملف المفتوح أو المحفوظ على الترتيب.



تحديد أنواع الملفات المعروضة من خلال خاصية **Filter**

لأدوات مربعات الملفات خاصية تحدد أنواع الملفات التي سيظهرها المربع عند عرضه وإن لم تحدها فإن القيمة الافتراضية لها هي كل الملفات، ولكننا من الممكن أن نظهر ملفات ذات امتداد معين مثل **txt** أو **bmp** مثلاً. تعمل هذه الخاصية كمصفاة **Filter** ويمكن تحديدها عند التصميم بواسطة مربع الخصائص أو بواسطة الكود. وفي كلا الحالتين فإنها تكون نصاً مثل الآتي:

Text Files|*.txt

و هو يتكون من وصف لنوع الملفات ثم الرمز " | " ويسمى **pipe symbol** ثم الامتداد الخاص بهذا النوع. ويمكننا أيضاً تحديد عدة أنواع بفصل كل نوع بالرمز " | " السابق، كما في المثال التالي:

Text Files|*.txt|Word Documents|*.doc

لاحظ عند التحديد بالكود ضرورة استخدام علامات الاقتباس كالتالي:

SaveFileDialog1.Filter = "Text Files|*.txt|Word Documents|*.doc|Documents|*.doc"

لا تترك أي مسافة بين الرمز " | " والامتداد، سواء قبله أو بعده، وإلا لن يعمل الاختزال بصورة صحيحة.



مربع حوار الخط " **Font Dialog Box** "

ربما ترغب من وقتٍ لآخر إتاحة الفرصة أمام مستخدمى برنامجك لتغيير خطوط أحد الكائنات الموجودة بالتطبيق، وهذا ما يمكنك منه أداة مربع الخطوط **FontDialog** التي يتم استخدامها باستدعاء الوظيفة **ShowDialog()** كما كنا نفعل مع مربعات الملفات. وبمجرد اختيار المستخدم للخط، يتم تخزين هذا الاختيار داخل الخاصية **Font** التي يمكنك أيضاً من خلالها تحديد اسم الخط الذى يظهر تلقائياً بمجرد

فتح مربع الخط **Font**. يمكنك أداء ذلك داخل مربع الخصائص أثناء مرحلة التصميم أو من خلال الكود كما في العبارة التالية:

FontDialog1.Font = lblTest.Font

والذى يتم فيه تخصيص الخط المستخدم فى أداة العنوان **lblTest** إلى مربع **Fonts**. للتعرف على طريقة إظهار واستخدام مربع الخط **Font**، تابع معنا الخطوات الآتية:

١. من مربع أدوات التحكم، انقر الأداة **FontDialog**  نقرًا مزدوجاً لإضافتها إلى النموذج حيث يتم وضع الأداة أسفل النموذج أيضاً مع تخصيصها بالاسم **FontDialog1**.

٢. قم بإضافة زر جديد إلى النموذج بعنوان **Font Dialog** مع تغيير اسمه إلى اسم معبر وليكن **btnFont**.

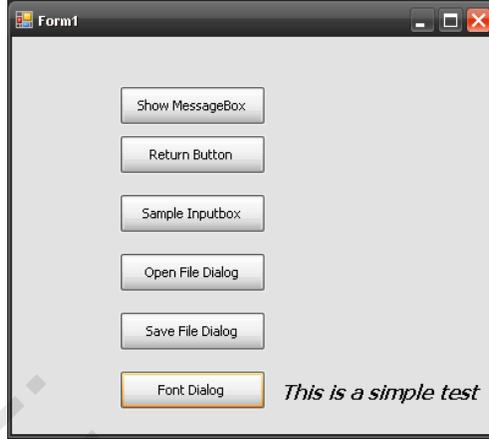
٣. قم بإضافة أداة عنوان إلى النموذج بالقرب من الزر الجديد بعنوان **This is a simple test** واسم **lblFont**.

٤. انقر الزر الجديد نقرًا مزدوجاً ثم قم بإدخال الكود التالي داخل إجراء احتواء الحدث **btnFont_Click**:

```
FontDialog1.Font = lblFont.Font
FontDialog1.ShowColor = True
FontDialog1.ShowDialog()
lblFont.Font = FontDialog1.Font
```

٥. حيث قمنا بتخصيص الخط الحالى لأداة العنوان كحالة ابتدائية للمربع **Font** كما قمنا بتخصيص القيمة **True** للخاصية **ShowColor** لتمكين المستخدم من اختيار اللون المناسب للخط. وبعد ذلك تم استدعاء الوظيفة **ShowDialog()** لعرض المربع الحوارى ثم تغيير خط أداة العنوان بالخط الذى يقوم المستخدم باختياره.

٦. قم بتشغيل التطبيق ثم انقر الزر الجديد، يتم فتح المربع الحوارى الشهير **Font**.
٧. قم باختيار الخط الذى يناسبك ثم انقر زر **Ok**، تلاحظ تغيير خط أداة العنوان إلى الإعدادات التى قمت بتحديددها (انظر شكل ١٢-٨).



شكل ١٢-٨ تغيير إعدادات الخط المستخدم من خلال المربع الحوارى Font.

خصائص أداة مربع الخط

تحتوى أداة مربع الخط FontDialog على العديد من الخصائص التى يمكنك تعيينها من داخل مربع الخصائص أو من داخل الكود. يوضح جدول ١٢-٦ التالى أهم هذه الخصائص واستخدام كل منها.

جدول ١٢-٦ خصائص أداة مربع الخط

| الخاصية | الاستخدام |
|--------------------|---|
| AllowVectorFonts | تستخدم لإظهار الخطوط ذات النمط Vector إذا كانت قيمتها True (القيمة الافتراضية). |
| AllowVerticalFonts | تستخدم لإظهار الخطوط الموجهة أفقياً أو رأسياً إذا كانت قيمتها True (القيمة الافتراضية). أما إذا كانت قيمتها False فيتم إظهار الخطوط الموجهة أفقياً فقط. |
| Color | تستخدم فى تعيين لون الخط المختار |
| FixedPitchOnly | إذا كانت True يتم إظهار الخطوط ذات النمط |

| الخاصية | الاستخدام |
|-------------------------|--|
| | Fixed-pitch فقط، والقيمة الافتراضية لها هي .False . |
| FontMustExist | لا تسمح للمستخدم بتعيين خط غير موجود إذا كانت قيمتها True (القيمة الافتراضية) |
| MaxSize, MinSize | تعيين أقل وأقصى عدد من النقاط (حجم) يمكن للمستخدم اختياره. |
| ShowApply | تستخدم في إظهار الزر Apply إذا كانت قيمتها True إلا أن القيمة الافتراضية لها هي .False . |
| ShowColor | تستخدم في إظهار الألوان لتمكين المستخدم من تغيير لون الخط إذا كانت قيمتها True ، إلا أن القيمة الافتراضية لها هي .False . |
| ShowHelp | حينما تكون قيمة هذه الخاصية True يتم إظهار زر مساعدة بالمربع الحوارى ومن ثمّ يمكنك كتابة كود المساعدة داخل الحدث HelpRequested . |

المربع الحوارى **Color**

يتيح مربع الحوار "لون" **Color Dialog Box** اختيار أحد الألوان لغرض معين مثل تغيير لون الخلفية أو لون كائن ما (انظر شكل ١٢-٩). استخدام هذا المربع يشبه مثيله المربع الحوارى **Font** إلا أنه يستخدم الأداة **ColorDialog** لاختيار الألوان باستدعاء الوظيفة **ShowDialog()** كما كنا نفعل مع مربعات الملفات. وبمجرد اختيار المستخدم للون، يتم تخزين هذا الاختيار داخل الخاصية **Color** ويكون حينئذٍ لديك الخيار فى استخدام أحد الألوان الموجودة من قبل أو إنشاء لوناً مخصصاً.



شكل ١٢-٩ المربع الحوارى Color.

١. للتعرف على طريقة إظهار واستخدام مربع اللون **Color**، تابع معنا الخطوات الآتية:
من مربع أدوات التحكم، انقر الأداة **ColorDialog**  نقرأ مزدوجاً لإضافتها إلى النموذج حيث يتم وضع الأداة أسفل النموذج أيضاً مع تخصيصها بالاسم **ColorDialog1**.
٢. قم بإضافة زر جديد إلى النموذج بعنوان **Color Dialog** مع تغيير اسمه إلى اسم معبر وليكن **btnColor**.
٣. انقر الزر الجديد نقرأ مزدوجاً ثم قم بإدخال الكود التالي داخل إجراء احتواء الحدث **btnColor_Click**:

```
ColorDialog1.Color = lblFont.BackColor
ColorDialog1.ShowDialog()
lblFont.BackColor = ColorDialog1.Color
```

٤. حيث قمنا بتخصيص اللون الحالى لخلفية أداة العنوان كحالة ابتدائية للمربع **ColorDialog1** وبعد ذلك تم استدعاء الوظيفة **ShowDialog()** لعرض المربع الحوارى ثم تغيير لون خلفية أداة العنوان باللون الذى يقوم المستخدم باختياره من المربع الحوارى **Color**.
٥. قم بتشغيل التطبيق ثم انقر الزر الجديد، يتم فتح المربع الحوارى الشهير **Color**.
٦. قم باختيار اللون الذى يناسبك ثم انقر زر **Ok**، تلاحظ تغيير لون خلفية أداة العنوان إلى اللون الذى قمت بتحديدته.

خصائص الأداة ColorDialog

يحتوى مربع الخصائص المصاحب للأداة ColorDialog على مجموعة من الخصائص التى يمكنك استخدامها للتحكم فى عرض المربع الحوارى المصاحب Color. يحتوى جدول ١٢-٧ التالى على أهم هذه الخصائص واستخدام كلٍ منها.

جدول ١٢-٧ الخصائص الشهيرة للأداة ColorDialog

| الخاصية | الاستخدام |
|----------------|---|
| AllowFullOpen | تستخدم لتمكين المستخدم من تعريف الألوان المخصصة إذا كانت قيمتها True (القيمة الافتراضية). أما القيمة False فتتسبب فى تعطيل الزر Define Custom Colors. |
| AnyColor | تستخدم لإظهار جميع الألوان الأساسية المتاحة إذا كانت قيمتها True إلا أن القيمة الافتراضية لها هي False. |
| FullOpen | تستخدم لإظهار أزرار أدوات الألوان المخصصة بمجرد فتح المربع الحوارى للمرة الأولى إذا كانت قيمتها True إلا أن القيمة الافتراضية لها هي False. |
| SolidColorOnly | إذا كانت قيمتها True يستطيع المستخدم اختيار الألوان المصممة فقط إلا أن القيمة الافتراضية لها هي False. |

بجانب المربعات الحوارية التى تعرضنا لها بالشرح حتى الآن، هناك أداتان تنتميان للتصنيف CommonDialog الذى تنتمى إليه أيضاً جميع المربعات الحوارية السابقة وهما الأداة PrintDialog المستخدمة للتحكم فى الطباعة والطباعة والأداة PageSetupDialog المستخدمة للتحكم فى إعداد الصفحة. وسوف نتعرض لهما بالتفصيل عند حديثنا على إنشاء التقارير من قواعد البيانات فى الجزء الثالث من هذه المجموعة.



إنشاء المربعات الحوارية الخاصة

رغم الاستخدام الواسع للمربعات الحوارية السابقة إلا أنها تعجز أحياناً عن تلبية حاجات خاصة للمبرمج، لكي تدرك ذلك عليك أن تتذكر أنه في كل الأنواع السابقة لم يكن ممكناً تغيير العناوين التي تظهر على الأزرار كما لم يكن ممكناً إضافة زر آخر. بدون حصر لأوجه القصور نقول أنه إن لم تجد ما تحتاجه في كل ما سبق فعليك تصميم المربع الحوارى الخاص بك. وسنقوم فيما يلي بتصميم مربع حوارى شهير لإدخال اسم المستخدم وكلمة المرور Username & Password مثل المستخدم للدخول إلى شبكة الإنترنت (إذا لم يكن هناك إنترنت مجاني بالطبع). كما ترى لا يوجد أي مربع حوارى يفيدنا في هذه الحالة لذا علينا إنشاؤه بالكامل وسيكون ذلك نموذجاً جيداً لإعداد المربعات الحوارية الأكثر تعقيداً.

إعداد نموذج المربع الحوارى

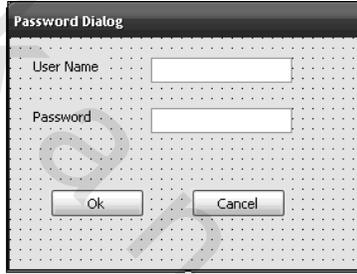
يبدأ إنشاء المربع الحوارى بإنشاء النموذج الذي يحتويه، لتنفيذ ذلك اتبع الآتي:

١. قم بإضافة نموذج جديد إلى التطبيق الحالى. لأداء ذلك، افتح قائمة **Project** من شريط القوائم ثم اختر **Add Windows Form** من القائمة المنسدلة، حيث يظهر المربع الحوارى **Add New Item**. قم بتعيين اسم مميز للنموذج الجديد وليكن **frmDialog** ثم انقر زر **Ok** لإضافة النموذج إلى مستكشف الحل.
٢. تأكد من ظهور مربع الخصائص وإلا اضغط مفتاح **F4** من لوحة المفاتيح لإظهاره.
٣. قم بتغيير عنوان النموذج من الخاصية **Text** باستخدام اسم معبر وليكن **Password Dialog**.
٤. قم بتغيير الخاصية **FormBorderStyle** إلى **Fixed Dialog** ومعناها تثبيت حجم النموذج وعدم تمكين المستخدم من تغيير حجمه .
٥. قم بتغيير قيمة الخاصية **ControlBox** داخل المجموعة **Windows Style** إلى **False** وكذلك الحال بالنسبة للخصائص **MaximizeBox** و **MinimizeBox** ومعناها عدم إظهار مربع قائمة التحكم في النموذج الذي يظهر عادةً في أقصى اليسار من شريط عنوان النموذج عند تنفيذ التطبيق ولا الأزرار التى تظهر بالركن الأيمن من النموذج.

٦. غير الخاصية **StartPosition** إلى **Center Screen** ومعناها إظهار المربع في البداية وسط الشاشة.

إضافة الكائنات (الأدوات) إلى النموذج

بعد إنشاء النموذج أضف مربعي نص **Textboxes** ، اختر للأول اسم **txtName** وللثاني اسم **txtPassword** وعنوان (**Label**) لكل منهما ، وزري أمر (**Buttons**) . اختر للأول اسم **btnOK** وللثاني اسم **btnCancel** كما تعلمت في الفصول السابقة. استخدم المفاهيم التي تعلمتها سابقا في وضع وضبط الكائنات وتغيير عناوينها لتبدو النافذة كما في شكل ١٢-١٠.



شكل ١٢-١٠ نموذج المربع بعد إضافة عناصر التحكم إليه.

يبقى تغيير بسيط في مربع **txtPassword** وذلك بتخصيص الحرف * لخاصية **PasswordChar** داخل المجموعة **Behavior** بمربع الخصائص وذلك كي يظهر الرمز * بدلا من الأحرف التي يكتبها المستخدم تأميناً لكلمة السر.

ماذا سيعرض المربع؟

يجب أن نتأكد من وضع كلمة افتراضية لكلا المربعين ("الاسم"، "كلمة المرور") كما يجب أن نتأكد بعد إغلاق المربع من معرفة ما أدخله المستخدم. يجب أيضاً إغلاق المربع عند النقر على أي زر من زرّي الأمر.

ولكن كيف سنتمكن من تمرير البيانات من وإلى النموذج. بما أن النموذج كائن ، فيمكننا تعريف خصائص جديدة لهذا الكائن تمثل ما نريد تمريره ، وذلك باستخدام الكلمة **Public** قبل المتغير المراد تعريفه.

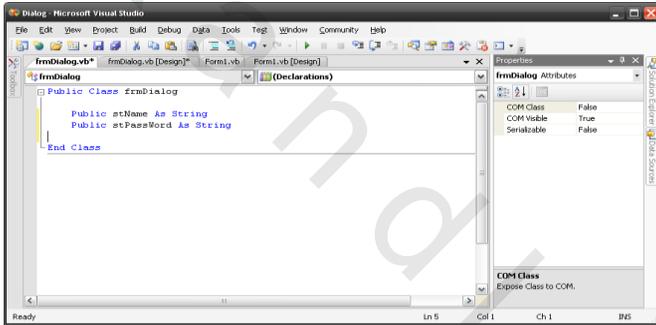
سنحتاج متغيرين هما **stName** و **stPassword** لتمرير الاسم وكلمة المرور على

الترتيب، ويمكننا أيضاً تعريف متغير لمعرفة اختيار المستخدم من الأزرار إلا أننا سنحاكي مربع الإدخال السابق شرحه وذلك بأن نرجع قيمة النص الخالي عند نقر **Cancel** ونرجع القيمة المدخلة عند نقر زر **OK** .

لتعريف المتغيرين، انقر النموذج الجديد نقرأ مزدوجاً لفتح نافذة الكود. واختر **Declaration** من المربع المنسدل الأيمن (مربع الأحداث) ثم اكتب السطرين الآتيين داخل التصنيف **frmDialog** (انظر شكل ١٢-١١):

```
Public stName as String
Public stPassWord as String
```

كلمة **Public** تتيح التعامل مع هذه المتغيرات من خارج النموذج أي من إجراءات النموذج الآخر في تطبيقنا مثلاً، فيمكن قراءة قيمة **stName** كالتالي باستخدام التسمية المنقوطة السابق توضيحها.



شكل ١٢-١١ تعريف المتغيرات داخل تصنيف النموذج.

عند إظهار المربع سيقوم المبرمج بتحديد قيمتين لهذين المتغيرين ثم يظهره ثم بعد ذلك يقرأ هاتين القيمتين، وعليه يجب على المربع أن يعكس القيم الممررة من قبل المستخدم على مربعات النصوص، يتم ذلك من خلال الإجراء الحداثي **Activate** الخاص بالنموذج حيث نحدد فيه قيم مربعات النصوص. من نافذة الكود اختر **frmDialog Events** من مربع الكائنات ثم اختر **Activated** من مربع الأحداث وقم بإدخال الكود التالي داخل إجراء الحدث **frmDialog_Activated** :

```
txtName.Text = stName
txtPassword.Text = stPassWord
```

يقوم هذا الإجراء بوضع القيم الافتراضية في مربعات النصوص. ثم عند إغلاق المربع بالزر

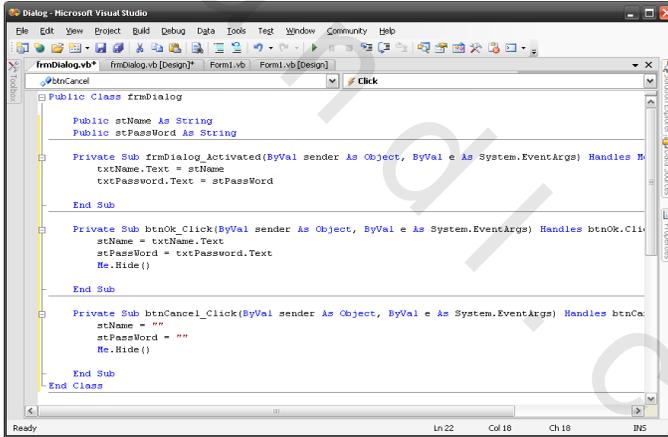
OK يتم نقل قيم المربعات إلى المتغيرات وذلك في الإجراء الحداثي Click للزر btnOK. من نافذة الكود، اختر btnOk من مربع الكائنات وClick من مربع الأحداث ثم قم بإدخال الكود التالي داخل إجراء الحدث btnOk_Click:

```
stName = txtName.Text
stPassWord = txtPassword.Text
Me.Hide
```

بينما عند إغلاق المربع بالزر Cancel يتم وضع القيمة "" في المتغيرين. من نافذة الكود، اختر btnCancel من مربع الكائنات وClick من مربع الأحداث ثم قم بإدخال الكود التالي داخل إجراء الحدث btnCancel_Click:

```
stName = ""
stPassWord = ""
Me.Hide
```

يجب أن تكون إجراءات نافذة الكود الخاصة بالنموذج الآن مثل شكل ١٢-١٢.



شكل ١٢-١٢ إجراءات نموذج مربع كلمة المرور.

استدعاء المربع الخاص من التطبيق

بدا واضحا الآن كيف سنستخدم هذا المربع من داخل التطبيق، مثلا أضف زر أمر جديد إلى النموذج الأول بعنوان Log On واسم btnLogon ثم قم بإدخال الكود التالي في الإجراء الحداثي btnLogon_Click:

```
Dim ThisDlg As New frmDialog()
```

```

ThisDlg.ShowDialog()
If ThisDlg.stName = "waleed" And ThisDlg.stPassWord = "root"
Then
    ThisDlg.Close()
    MessageBox.Show("Information Ok")
Else
    ThisDlg.Close()
    MessageBox.Show("Bad Information")
End
End If

```

حيث يتم أولاً إنشاء حالة من النموذج frmDialog() داخل النموذج الحالي وبعد ذلك يتم عرض النموذج من خلال الوظيفة ShowDialog(). لاحظ أننا نقارن الاسم وكلمة المرور بـ "waleed" و "root" على الترتيب ويمكنك تغييرهما هنا أو المقارنة بمتغيرات بدلا من الثوابت حتى يمكن تغييرها أثناء التشغيل.

يبدو شكل المربع الحوارى "كلمة السر" الذي أنشأناه أثناء تشغيل التطبيق، وبعد إدخال اسم المستخدم وكلمة المرور وقبل نقر أي من زري الأمر بالموافقة أو الرجوع عنها مثل شكل ١٢-١٣ التالى.



شكل ١٢-١٣ المربع الحوارى "كلمة المرور" الذي أنشأناه أثناء تشغيل التطبيق.