الفصل العشرون
تصميم واجهات متعددة المستندات

نشرح فى هذا الفصل التطبيقات التى تستخدم واجهات متعددة المستندات، كيفية إنشائها وخصائص نوافذها، حيث نقوم بإنشاء تطبيق متكامل يصلح للاستخدام فى تطبيقات أكبر فيما بعد كما يتعرض الفصل تقريباً لجميع المهام التى شرحناها فى الكتاب حتى الآن، لذا فهو يعتبر تطبيق شامل على ما تم دراسته بالكتاب.

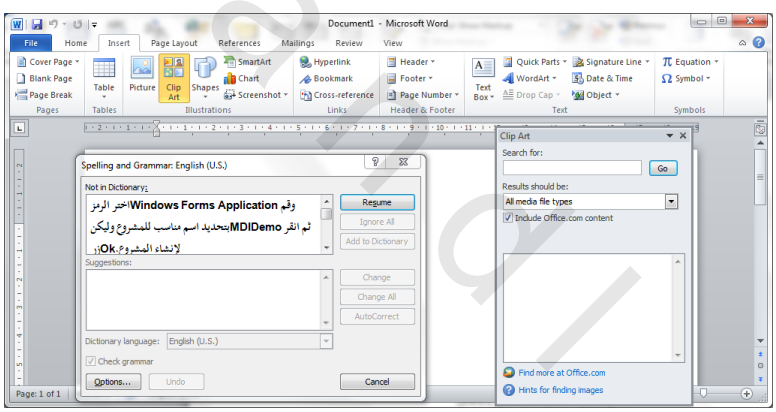
بانتهاء هذا الفصل، ستتعرف على:

- الاختلاف بين التطبيقات ذات الواجهة متعددة النماذج، وتلك التى تستخدم واجهات قياسية.
- مواصفات النوافذ الأم والنوافذ الوليدة.
- إنشاء النافذة الأم وتعيين خصائصها.
- إنشاء قالب النافذة الوليدة وتعيين خصائصها.
- استخدام القوائم مع كل من النافذة الأم والنافذة الوليدة.
- فتح الملفات والمستندات وحفظها وطباعتها وإغلاقها.

مقدمة إلى التطبيقات متعددة المستندات

أغلب البرامج التي قمنا بإنشائها والتي ستحتاجها غالباً ما تكون مكونة من عدة نماذج أو نوافذ منفصلة كما يظهر في شكل ٢٠-١ هذه النماذج يمكن تحريكها وتصغيرها وتكبيرها على انفراد، يعطي ذلك حرية للمستخدم إلا أنه لا توجد طريقة سهلة لتنظيم وضع هذه النماذج إذا أردنا، وكذلك تنظيم قوائمها في صورة مجموعات.

لأن النموذج (From) يظهر دائماً في إطار يسمى نافذة .فستجد في الشرح هنا الإشارة إليه بكلمة "نافذة"أو كلمة "نموذج" أو كلمة "نافذة نموذج" وكلها مترادفات للكلمة الإنجليزية Form.



شكل ٢٠-١ نموذج لأحد البرامج متعددة النوافذ

أحد البدائل لذلك هو استخدام الواجهة متعددة المستندات Multiple document interface (MDI) حيث توجد نافذة أم Parent تحوي كل أو معظم نوافذ البرنامجوالتي تكون حينئذ نوافذ وليدة Childs،يفيد هذا النظام في التحكم في كل النوافذ الوليدة معاً بالتحكم في النافذة الأم فقط، مثل التصغير والتكبير وما إلى ذلك من الخصائص الأخرى. كما يفيد في تحسين برنامجك في اتجاهين:

الأول: أن لديك حاوية واحدة للنوافذ، تتحرك محتوياتها معها أينما ذهبت، وهذا من شأنه أن يظهر برنامجك بصورة منظمة وأن يجعله ملتئم ذاتياً.

الثاني: أنه يسمح للمستخدم بالعمل في عدة نوافذ (وثائق) في آن واحد، وهذا من شأنه أن يزيد إنتاجية المستخدم، ويريحه أثناء العمل.

إذا سبق لك التعامل مع برامج مثل Microsoft Word أو Microsoft Excelفبالتأكيد تعرف ذلك، حيث يمكنك فتح العديد من المستندات في نفس الوقت،كما يمكنك تصغير كل ذلك بتصغير النافذة الأم.لو لم تتعامل مع أي من هذين البرنامجين فيكفي تعاملك مع بيئة تطوير Visual Studio حيث أنها ذات واجهة متعددة المستندات بالفعل.

يعرض شكل ٢٠-٢ نموذج من التطبيقات ذات الواجهة متعددة المستندات.

شكل ٢٠-٢ مثال لأحد البرامج متعددة المستندات.

مواصفات النوافذ الأم MDI Parent Form

النافذة الأم أو اختصاراً النافذة MDI هي حاوية لكافة النوافذ الوليدة. وتتميز بعدة مواصفات عن النوافذ العادية تتحكم في سلوكهاوهي:

- تعمل النافذة الأم كحاوية للنوافذ الوليدة، لذا يتم رسم النوافذ الوليدة داخل نطاق النافذة الأم.

- تقوم النافذة الأم تلقائياً بتوفير أشرطة التمرير إذا تم وضع واحدة أو أكثر من النوافذ الوليدة خارج حدود النافذة الأم.
- يتم تمثيل النافذة الأم وجميع نوافذها الوليدة برمز واحد فقط على شريط المهام. فإذا تم تقليص النافذة الأم ثم أعيد تكبيرها مرةً أخرى يتم إظهار النوافذ الوليدة بالحجم والمكان التى كانت عليه قبل عملية التقليص.

مواصفات النوافذ الوليدة MDI child forms

تتميز النوافذ الوليدة بعدة مواصفات هى:

- النوافذ الوليدة يمكن إظهارها داخل إطار النافذة الأم، ولا يمكن تحريكها إلى خارجها.
- عند تصغير نافذة وليدة، يظهر الرمز الخاص بها في النافذة الأم وليس في شريط المهام.
- عند تكبير Maximizing النافذة الوليدة تملأ مساحة العمل في النافذة الأم تماماً، كذلك سيحتوي العنوان في النافذة الأم على اسمها بجانب عنوان النافذة الوليدة.
- عند تكبير إحدى النوافذ الوليدة يتم تكبير كافة النوافذ الوليدة تبعاً لذلك.

إنشاء تطبيق بسيط متعدد المستندات

نشرح هنا كيفية إنشاء تطبيق بسيط متعدد المستندات، يمكن استخدامه كأساس لأي تطبيق حقيقي تريد إنشاؤه، تتكون التطبيقات متعددة المستنداتغالباً من نافذة أم واحدة ونافذة وليدة واحدة على الأقل.

إنشاء النافذة الأم

لإنشاء المشروع الجديد ومن ثمَّ النافذة الأم، تابع معنا الخطوات الآتية:

١. افتح قائمة File من شريط القوائم ثم اختر NewProject من القائمة المنسدلة، يظهر المربع الحوارى المعتاد New Project.

٢. اختر الرمز Windows Forms Application وقم بتحديد اسم مناسب للمشروع وليكن MDIDemo ثم انقر زر Ok لإنشاء المشروع.

٣. انقر النموذج **Form1.vb** داخل نافذة مستكشف الحل ثم أعد تغيير اسمه إلى اسم معبر وليكن **frmParent.vb** لأننا سنستخدمه كنافذة أم للتطبيق.

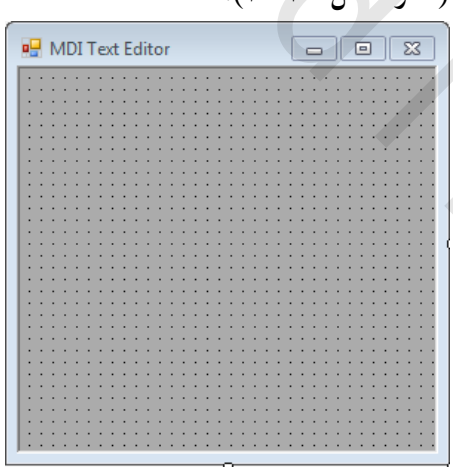
٤. انقر النموذج بزر الفأرة الأيمن ثم اختر **View Code** من القائمة الموضعية، تظهر نافذة الكود الخاصة بالنموذج.

٥. تأكد من تغيير اسم التصنيف **Form1** إلى **frmParent** وإلا قم بتغييره.

٦. انقر اسم المشروع بزر الفأرة الأيمن ثم اختر **Properties** من القائمة الموضعية ثم تأكد من اختيار **frmParent** بمربع السرد **Startup form** بالتبويب **Application** وإلا قم بتغيير هذه القيمة.

٧. انقر النموذج لتنشيطه ثم قم بتغيير الخاصية **Text** الخاصة بالنموذج إلى نص معبر وليكن **MDI Text Editor**.

٨. قم بتغيير الخاصية **IsMDIContainer** الخاصة بالنموذج إلى **True** وهذا يعنى أن هذا النموذج عبارة عن نافذة أم ودلالة ذلك ظهور خلفية النموذج بلون داكن داخل مصمم النماذج (انظر شكل ٢٠-٣).



شكل ٢٠-٣ تعمل النافذة الأم كحاوية للنوافذ الوليدة.

اعتبارات القوائم

كما تعلمت فى الفصل السابعمن هذا الكتاب، يمكنك إضافة أداة القائمة **MenuStrip Control** لإضافة خيارات القوائم إلى النماذج. ولعلك تتساءل الآن عن عدد أشرطة القوائم التى يمكنك إضافتها إلى تطبيق متعدد المستندات. وفى الواقع فإن **Visual Basic 2012** يحتوى على مجموعة من الخصائص التى تتحكم فى عدد هذه القوائم حينما تحتوى كل من النافذة الوليدة والنافذة الأم على شريط للأدوات. وفيما يلى نقوم بإلقاء الضوء على هذه الخصائص.

- الخاصية ***MergeAction***

يحتوى كل عنصر داخل القائمة على الخاصية **MergeAction**التى تحدد سلوك هذا العنصر عند دمجه مع عنصر مطابق بقائمة أخرى. يوضح جدول ٢٠-١ التالى القيم الممكنة لهذه الخاصية.

جدول ٢٠-١ قيم الخاصية **MergeAction**

القيمة	الاستخدام
Append	يتم إضافة عنصر القائمة إلى نهاية مجموعة عناصر القائمة التى يتم دمجه بها بغض النظر عن وجود تطابق من عدمه وهذه هى القيمة الافتراضية.
Insert	يتم مزج العناصر المتفرعة من هذا العنصر مع العناصر المتفرعة من القائمة التى يتم دمجه بها وقبل العنصر المطابق مباشرة
Remove	لا يتم تضمين هذا العنصر فى القائمة الناتجة من عملية الدمج
Replace	يتم استبدال هذا العنصر بالعنصر المطابق فى نفس المكان بالقائمة الناتجة من عملية الدمج
MatchOnly	يتم إجراء عملية التطابق مع عدم اتخاذ أى إجراءات

- الخاصية ***MergeIndex***

إذا أردت دمج أحد عناصر القائمة مع قائمة أخرى، قم بتحديد موقع العنصر فى القائمة الثانية من خلال الخاصية **MergeIndex** التى تأخذ القيم **0** و**1** وهكذا.

إنشاء قائمة النافذة الأم

سنقوم فيما يلى باستكمال إنشاء التطبيق الذى بين أيدينا حيث نقوم بإضافة القائمة التى تظهر بالنافذة الأم. تابع معنا الخطوات الآتية:

١. قم بإدراج أداة القائمة MenuStripمن مربع الأدوات إلى النموذج ثم قم بإنشاء قائمتين بالمستوى العلوى كما يلى:

الخاصية Name	الخاصية Text
mnuFile	File
mnuWindow	Window

٢. قم بإنشاء أربعة عناصر داخل قائمة File كما يلى:

الخاصية Name	الخاصية Text
mnuFileNew	New
mnuFileOpen	Open
mnuFileSep	–
mnuFileExit	Exit

٣. قم بإنشاء أربعة عناصر أيضاً داخل القائمة Window كما يلى:

الخاصية Name	الخاصية Text
mnuWindowTileH	Tile Horizontal
mnuWindowTileV	Tile Vertical
mnuWindowCascade	Cascade
mnuWindowSep	-

٤. انقر عنصر القائمة mnuFile ثم قم بتخصيص القيمة Insert للخاصية MergeAction وهذا من شأنه دمج عناصر قائمة النافذة الوليدة مع عناصر قائمة النافذة الأم.

٥. قم بتخصيص القيم 98 و99 للخاصية MergeIndex لعنصرى القائمة mnuFileSep وmnuFileExit على الترتيب وذلك كى نضمن ظهور العنصر

Exit فى زيل القائمة بعد الفاصل مباشرةً.

٦. لتعيين كود العنصر Exit، انقر عنصر القائمة Exit نقراً مزدوجاً ثم قم بإدخال الكود التالى داخل حدث النقر mnuFileExit_Click:

```
Me.Close()
```

إنشاء قالب النافذة الوليدة

بعد أن قمنا بإنشاء النافذة الأم، سنقوم فيما يلى بإنشاء قالب النافذة الوليدة، حيث سنقوم بإنشاء تصنيف النافذة الوليدة أثناء التصميم على أن نقوم بإنشاء التصنيفات الأخرى أثناء التشغيل تبعاً لمتطلبات المستخدم. لإنشاء قالب النافذة الوليدة، تابع معنا الخطوات الآتية:

١. افتح قائمة Project من شريط القوائم ثم اختر Add Windows Form من القائمة المنسدلة، يظهر المربع الحوارى Add New Item.

٢. قم بتعيين اسم مناسب للنموذج وليكن frmChild ثم انقر زر Add، يتم إضافة النموذج الجديد إلى المشروع.

٣. قم بإدراج أداة قائمة من مربع الأدوات إلى النموذج ثم قم بإنشاء القائمة File وعناصرها الفرعية كما يلى:

الخاصية Name	الخاصية Text
mnuFile	File
mnuFileSave	Save
mnuFileSaveAs	Save As
mnuFileClose	Close
mnuFileSep	-
mnuFilePrint	Print

٤. انقر عنصر القائمة mnuFile ثم قم بتخصيص القيمة Replace للخاصية MergeAction وهذا من شأنه دمج عناصر قائمة النافذة الوليدة مع عناصر قائمة النافذة الأم.

٥. قم بإدراج أداة مربع نص إلى النموذج وقم بتغيير اسم الأداة إلى txtMain.

٦. قم بتغيير خط مربع النص إلى خط صغير وليكن **Courier New** وذلك من خلال الخاصية **Font**.

٧. لأننا نرغب فى أن يملأ مربع النص النموذج بالكامل مهما تغير حجم النموذج، قم بتخصيص القيمة **True** للخاصية **MultiLine** والقيمة **Fill** للخاصية **Dock**.

٨. قم بتخصيص القيمة **Both** للخاصية **ScrollBars** حتى يظهر شريطا التمرير إذا زاد حجم النص الموجود بمربع النص عن المساحة المتاحة.

٩. قم بتخصيص القيمة **False** للخاصية **WordWrap** حتى يتمكن المستخدم من كتابة أى كمية من النص فى أى سطر من السطور بغض النظر عن عرض مربع النص نفسه.

تعيين خصائص النافذة الوليدة

نحتاج من النافذة الوليدة أن تحتوى على خاصيتين يتم تبادلهما مع النافذة الأم وهما:

- الخاصية **TextSaved** وهى عبارة عن متغير منطقى عام يحدد مدى حفظ النص الموجود بالنافذة الوليدة من عدمه ويأخذ إحدى القيمتين **True** أو **False**. قم بكتابة السطر التالى بنافذة كود النموذج **frmChild** وقبل السطر **End Class** مباشرةً:

```
Public TextSavedAs Boolean
```

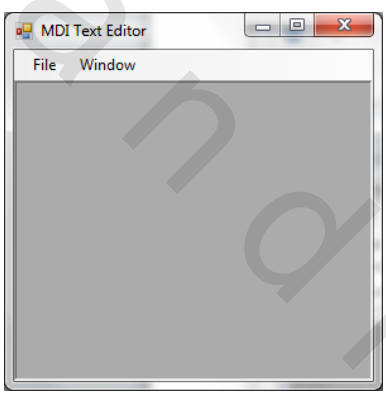
- الخاصية **FileName** وتحتوى على اسم الملف ومساره وهو الملف الذى يتم حفظ محتويات النافذة الوليدة بداخله. وإذا لم يتم حفظ الملف بعد، يتم تخصيص القيمة " " لهذه الخاصية. ولأن هذه الخاصية بها شئٌ من التعقيد مقارنةً بالخاصية السابقة، سنقوم بتعريفها من خلال إجراء خاصية. قم بإدخال الكود التالى أسفل الخاصية الأولى:

```
Private sFileName As String
   Public Property FileName() As String
      Get
         Return sFileName
      End Get
Set(ByVal Value As String)
sFileName = Value
         If Trim(sFileName) = " " Then
```

```
Me.Text = "Untitled"
        Else
Me.Text = Mid(sFileName, InStrRev(sFileName,
"\") + 1)
        End If
      End Set
   End Property
```

وهذا الكود كما ترى لا يحتاج إلى المزيد من التوضيح، حيث يتم تخزين اسم الملف داخل المتغير **sFileName**. فإذا كانت قيمته خالية، يتم إظهار كلمة **Untitled** على شريط عنوان النموذج وإلا يتم إظهار اسم الملف ومساره كاملاً.

قم الآن بتشغيل التطبيق، تحصل على النافذة الأم فقط محتويةً على القائمتين الرئيسيتين (انظر شكل ٢٠-٤).



شكل ٢٠-٤ النافذة الأم محتويةً على شريط القوائم المبسط.

إظهار النافذة الوليدة

الخطوة التالية تتضمن إنشاء وإظهار حالة جديدة من التصنيف **frmChild** أو بمعنى آخر إظهار نافذة وليدة جديدة داخل النافذة الأم. فمن المعتاد فتح نافذة وليدة (مستند) بدون عنوان بمجرد تشغيل البرنامج كما هو الحال فى العديد من البرامج الشهيرة كبرنامج **Excel** أو برنامج **Word**. وبالإضافة إلى ذلك هناك الخيار **New** داخل قائمة **File** والذى يتسبب فى إنشاء نافذة وليدة جديدة. ولأن هذه العملية يتم ندائها من أكثر من مكان بالبرنامج، فمن الطبيعى تمثيلها داخل إجراء عام قابل للاستدعاء من أى مكان.

إنشاء الإجراء ***NewChild***

سنقوم فيما يلى بإنشاء الإجراء **NewChild** (يمكنك بالطبع اختيار اسماً آخر) داخل تصنيف النافذة الأم **frmParent**. لأداء ذلك، افتح نافذة كود النموذج **frmParent** ثم قم بإدخال الكود التالى فى نهاية الكود وقبل السطر **End Class** مباشرةً:

```
Private Sub NewChild()
    Dim f As New frmChild()
f.MdiParent = Me
f.FileName = "Untitled "
f.TextSaved = True
f.Show()
  End Sub
```

وكما ترى يبدأ الإجراء بإنشاء متغير محلى **f** والذى يقوم بإنشاء حالة جديدة من النموذج الوليد **frmChild** وبعد ذلك يتم تخصيص النافذة الأم إلى الخاصية **MdiParent** وذلك من خلال كلمة **Me** وهى إحدى الكلمات الخاصة داخل **Visual Basic** والتى تعنى التصنيف الحالى ومن ثمَّ النموذج **frmParent**. وبعد ذلك يتم تخصيص القيمة **"Untitled "** لاسم الملف حتى يظهر العنوان **Untitled** وكذا تخصيص القيمة **True** للخاصية **TextSaved**التى قمنا بإنشائها وأخيراً يتم إظهار النموذج من خلال الوظيفة **()Show**.

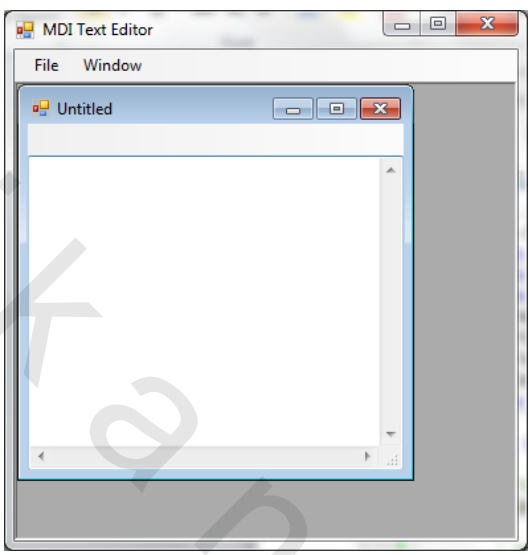
عرض نافذة وليدة بمجرد تشغيل التطبيق

إذا قمت بتشغيل التطبيق فى هذه اللحظة، تحصل على النافذة الأم خالية من أى نوافذ وليدة (راجع شكل ٢٠-٤). وكما ذكرنا ينبغى إظهار نافذة وليدة بمجرد تشغيل البرنامج. لذا يجب استدعاء الإجراء **NewChild** من داخل الإجراء **New** الخاص بالنافذة الأم. لأداء ذلك، تأكد أنك داخل نافذة كود النموذج **frmParent** ثم اختر **frmParent** من قائمة الكائنات إن لم تكن مختارةً بالفعل واختر **New** من قائمة الأحداث ثم قم بإدخال السطر التالى فى نهاية الإجراء **()New**وقبل السطر **End Sub** مباشرةً:

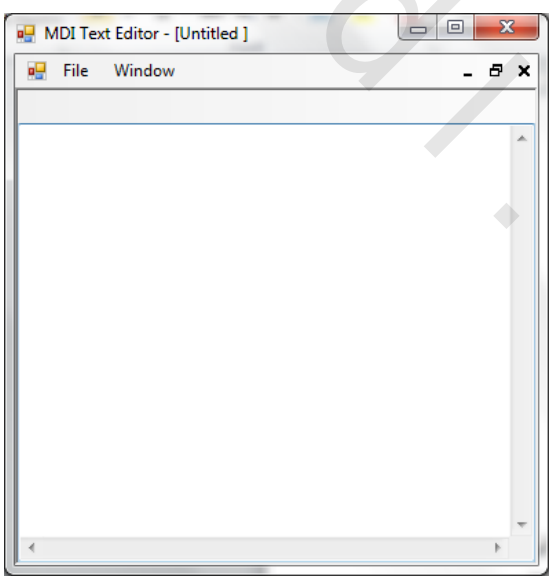
```
NewChild()
```

والآن قم باختبار التطبيق بعد هذه التعديلات. قم بتشغيل التطبيق، تحصل على نافذة وليدة بعنوان **Untitled** داخل النافذة الأم التى تحتوى على شريط القوائم (انظر شكل ٢٠-٥).

قم بتكبير النافذة الوليدة كى تملأ مساحة النافذة الأم بالكامل ولاحظ دمج العنوانين بشريط عنوان النافذة الأم، حيث يظهر عنوان النافذة الوليدة بين قوسين (انظر شكل ٢٠-٦).



شكل ٢٠-٥ تظهر نافذة وليدة بعنوان **Untitled** داخل النافذة الأم بمجرد تشغيل التطبيق.



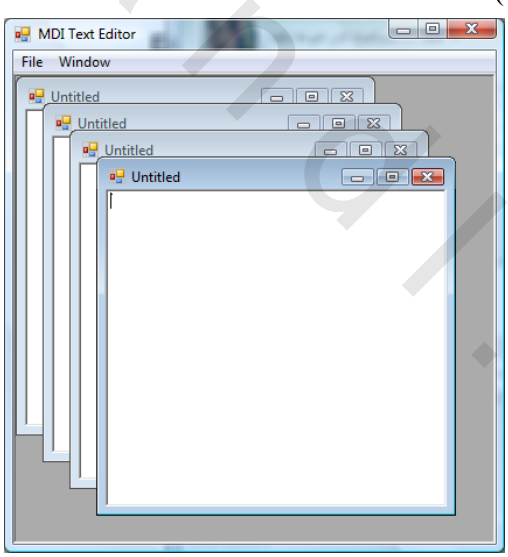
شكل ٢٠-٦ بعد تكبير النافذة الوليدة يتم دمج عنوانها مع عنوان النافذة الأم.

تمكين المستخدم من إظهار نافذة وليدة جديدة

كما ذكرنا من قبل هناك حالتين نحتاج فيهما إلى إظهار نافذة وليدة جديدة، الأولى فى بداية تشغيل التطبيق وقد قمنا بإعدادها، والثانية حينما يقوم المستخدم باختيار **New** من قائمة **File** الموجودة بالنافذة الأم. لأداء ذلك، انقر خيار القائمة **New** نقراً مزدوجاً، ثم قم باستدعاء الإجراء **NewChild** داخل حدث النقر كما يلى:

```
Private Sub MnuFileNew_Click(sender As Object, e As
EventArgs) Handles MnuFileNew.Click
NewChild()
End Sub
```

والآن قم بتشغيل التطبيق، ثم افتح قائمة **File** واختر **New** من القائمة المنسدلة، تلاحظ إضافة نافذة وليدة جديدة. اختر **New** مرةً أخرى، تلاحظ إضافة نافذة أخرى وهكذا (انظر شكل ٢٠-٧).



شكل ٢٠-٧ إضافة نوافذ وليدة جديدة من خلال الخيار **New**.

فتح الملفات والمستندات

بالإضافة إلى إمكانية فتح نافذة وليدة خالية، يجب أن يتمكن المستخدم من فتح الملفات

والمستندات ووضعها داخل النوافذ الوليدة الجديدة وذلك من خلال أداة مربع فتح الملفات **OpenFileDialog** (راجع الفصل الثامن من هذا الكتاب). لأداء ذلك، تابع معنا الخطوات الآتية:

١. قم بإدراج الأداة **OpenFileDialog** إلى النافذة الأم ثم أعد تسميتها إلى **dlgOpen**.

٢. انقر الخيار **Open** داخل قائمة **File** بالنافذة الأم نقراً مزدوجاً ثم قم باستدعاء الوظيفة **()ShowDialog** لفتح المربع الحوارى **Open** كما يلى:

```
Private Sub mnuFileOpen_Click(sender As Object, e As
EventArgs) Handles mnuFileOpen.Click
dlgOpen.ShowDialog()
End Sub
```

٣. تحتوى الأداة **OpenFileDialog** على الحدث **FileOk** الذى يتم الوصول إليه بمجرد اختيار المستخدم لملف داخل المربع الحوارى **Open** ثم نقر زر **Open**. سنقوم فيما يلى بتعريف كود فتح الملف داخل مربع النص الموجود بالنافذة الوليدة داخل هذا الحدث. ولأننا نحتاج إلى استخدام بعض وظائف الإدخال والإخراج، فمن الضرورى إحضار المسمى **System.IO** إلى كود التصنيف **frmParent**.

٤. قم بكتابة السطر التالى قبل بداية التصنيف **frmParent**:

```
Imports System.IO
```

٥. اختر **dlgOpen** من مربع الكائنات ثم اختر **FileOk** من مربع الأحداث.

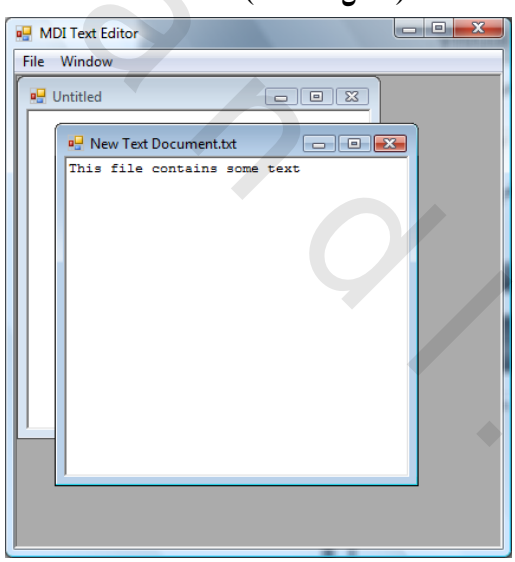
٦. قم بإدخال الكود التالى إلى الحدث **dlgOpen_FileOk**:

```
Private Sub dlgOpen_FileOk(sender As Object, e As
System.ComponentModel.CancelEventArgs) Handles
dlgOpen.FileOk
Dim f As New frmChild()
Dim nType As Integer = FreeFile()
Dim myFile As StreamReader
f.MdiParent = Me
myFile = File.OpenText(dlgOpen.FileName)
f.txtMain.Text = myFile.ReadToEnd
f.txtMain.Select(0, 0)
```

```
myFile.Close()
f.FileName = dlgOpen.FileName
f.Show()
End Sub
```

للتعرف على المزيد عن المسمى **System.IO** ووظائفه المختلفة، راجع الجزء الثانى من هذا الكتاب.

والآن قم بتشغيل التطبيق ثم افتح قائمة **File** واختر **Open** من القائمة المنسدلة، يظهر المربع الحوارى **Open**. اختر الملف النصى الذى ترغب فى إظهاره ثم انقر زر **Open**، يتم إغلاق المربع الحوارى وفتح الملف داخل مربع نص نافذة وليدة جديدة ويظهر اسم الملف على شريط عنوان هذه النافذة (شكل ٢٠-٨).



شكل ٢٠-٨ فتح ملف نصى داخل نافذة وليدة جديدة.

إدارة النوافذ الوليدة

تعطى معظم التطبيقات متعددة المستندات للمستخدم القدرة على ترتيب النوافذ الوليدة المفتوحة ترتيباً معيناً وذلك من خلال القائمة **Window** حيث يوجد ثلاثة ترتيبات شهيرة هى:

- الترتيب الأفقى **Tiled Horizontally** وفيه يتم وضع جميع النوافذ المفتوحة فى مستطيلات أفقية فوق بعضها البعض (انظر شكل ٢٠-٩).
- الترتيب الرأسى **Tiled Vertically** وفيه يتم وضع جميع النوافذ المفتوحة فى مستطيلات رأسية بجانب بعضها البعض (انظر شكل ٢٠-١٠).
- الترتيب المتتالى **Cascaded** وفيه يتم وضع جميع النوافذ فوق بعضها بدءاً من الركن الأيسر العلوى من النافذة الأم على أن يظهر شريط عنوان كل منها (انظر شكل ٢٠-١١).

وفى جميع الحالات يتم تخصيص حجم متساوى لجميع النوافذ المفتوحة. يمكنك أداء ذلك بسهولة تامة باستخدام الوظيفة **LayoutMdi()** مع استخدام أيٍ من المعاملات **Cascade** و**TileHorizontal** و**TileVertical**. لأداء ذلك، تابع معنا الخطوات الآتية:

١. انقر الخيار **Tile Horizontal** داخل القائمة **Window** بالنافذة الأم نقراً مزدوجاً ثم قم باستدعاء الوظيفة **LayoutMdi()** كما يلى:

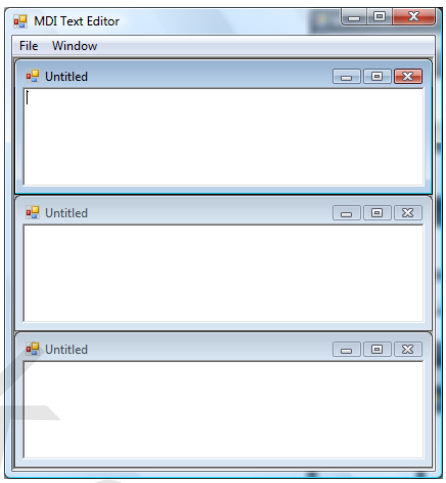
```
Private Sub mnuWindowTileH_Click(sender As Object, e As
EventArgs) Handles mnuWindowTileH.Click
Me.LayoutMdi(MdiLayout.TileHorizontal)
End Sub
```

٢. انقر الخيار **Tile Vertical** داخل القائمة **Window** بالنافذة الأم نقراً مزدوجاً ثم قم باستدعاء الوظيفة **LayoutMdi()** كما يلى:

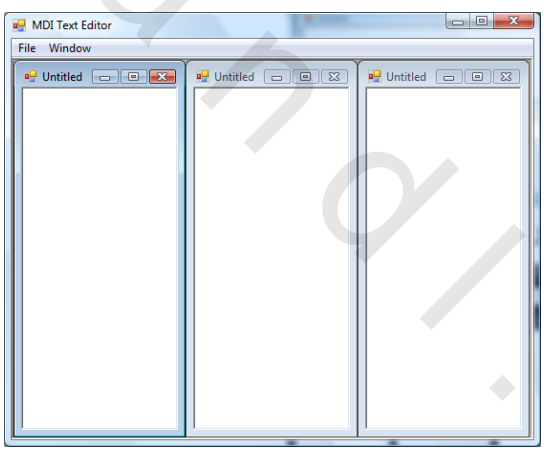
```
Private Sub mnuWindowTileV_Click(sender As Object, e As
EventArgs) Handles mnuWindowTileV.Click
Me.LayoutMdi(MdiLayout.TileVertical)
End Sub
```

٣. انقر الخيار **Cascade** داخل القائمة **Window** بالنافذة الأم نقراً مزدوجاً ثم قم باستدعاء الوظيفة **LayoutMdi()** كما يلى:

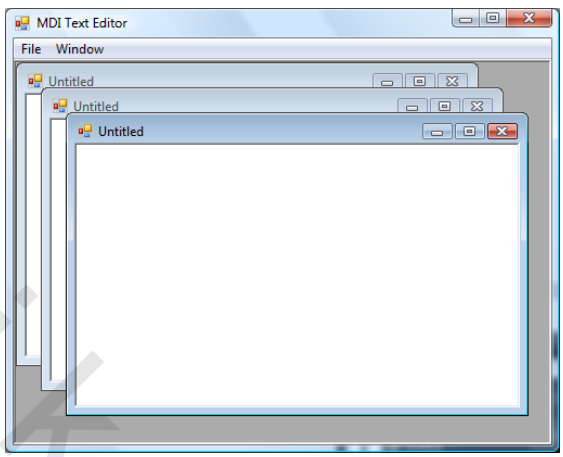
```
Private Sub mnuWindowCascade_Click(sender As Object, e As
EventArgs) Handles mnuWindowCascade.Click
Me.LayoutMdi(MdiLayout.Cascade)
End Sub
```



شكل ٢٠-٩ ترتيب النوافذ الوليدة أفقياً.



شكل ٢٠-١٠ ترتيب النوافذ الوليدة رأسياً.



شكل ٢٠-١١ الترتيب المتتالى للنوافذ الوليدة.

التبديل بين النوافذ المفتوحة

يمكنك التبديل ببين النوافذ المفتوحة عن طريق نقر أيٍ من هذه النوافذ لتنشيطها، كما يمكنك فتح قائمة **Window** التى تحتوى على عناوين جميع النوافذ المفتوحة ومن ثمَّ اختيار النافذة المطلوبة.

حفظ الملفات

رأينا فيما سبق أن فتح الملفات يتم من خلال النافذة الأم، أما حفظ هذه الملفات فيتم حقيقةً من خلال النافذة الوليدة نفسها وذلك من خلال الخيارين **Save** و**Save AS** ولا شك عزيزى القارئ أنك تعلم الفرق بينهما من دراستك لنظام التشغيل **Windows**. وقد قمنا فيما سبق بتعريف الخاصية **TextSaved** التى من خلالها نستطيع التعرف عما إذا كان الملف قد تم حفظه أم لا. قم بإدراج الأداة **SaveFileDialog** إلى النموذج الوليد وأعد تسميتها إلى **dlgSave**.

ولأننا نستطيع التعرف على وقت تغيير النص داخل مربع النص الموجود بالنافذة الوليدة من خلال الحدث **TextChanged**، إذاً من الضرورى تخصيص القيمة **False** لهذه الخاصية داخل هذا الحدث كما يلى:

```
Private Sub txtMain_TextChanged(sender As Object, e As
EventArgs) Handles txtMain.TextChanged
Me.TextSaved = False
End Sub
```

سنقوم فيما يلى بإنشاء الإجراء **SaveFile** والإجراء **SaveFileAs** وهى الإجراءات التى يتم استدعائها داخل حدث نقر الخيار **Save** والخيار **SaveAs** على الترتيب. تابع معنا الخطوات الآتية:

١. قم بكتابة السطر التالى داخل كود النموذج الوليد **frmChild** وقبل كود التصنيف نفسه:

```
Imports System.IO
```

٢. قم بإدخال الإجراء **SaveFile** التالى داخل التصنيف **frmChild** وقبل السطر **End Class** مباشرةً:

```
Private Sub SaveFile()
Dim myFile As StreamWriter
If sFileName = "" Then
SaveFileAs()
End If
myFile = File.CreateText(sFileName)
myFile.Write(txtMain.Text)
myFile.Close()
Me.TextSaved = True
End Sub
```

٣. قم بإدخال الإجراء **SaveFileAs** التالى داخل التصنيف **frmChild** وبعد الإجراء **SaveFile** مباشرةً:

```
Private Sub SaveFileAs()
dlgSave.FileName = Me.FileName
dlgSave.ShowDialog()
End Sub
```

٤. لتعيين كود الخيار **Save** داخل قائمة **File** بالنافذة الوليدة، انقر الخيار نقراً مزدوجاً ثم قم بإدخال الكود التالى:

```
Private Sub mnuFileSave_Click(sender As Object, e As
EventArgs) Handles mnuFileSave.Click
```

```
    If sFileName> "" Then
SaveFile()
    Else
SaveFileAs()
    End If
End Sub
```

٥. لتعيين كود الخيار Save As داخل قائمة File بالنافذة الوليدة، انقر الخيار نقراً مزدوجاً ثم قم بإدخال الكود التالى:

```
Private Sub mnuFileSave_Click(sender As Object, e As
EventArgs) Handles mnuFileSave.Click
SaveFileAs()
End Sub
```

٦. تحتوى الأداة SaveFileDialog على الحدث FileOk الذى يتم الوصول إليه بمجرد حفظ المستخدم لملف داخل المربع الحوارى Save ثم نقر زر Save. قم بتعريف كود حفظ الملف داخل هذا الحدث كما يلى:

```
Private Sub dlgSave_FileOk(sender As Object, e As
System.ComponentModel.CancelEventArgs) Handles
dlgSave.FileOk
    If e.Cancel Then
Me.FileName = dlgSave.FileName
SaveFile()
    End If
End Sub
```

التحكم فى المستندات أثناء إغلاق النوافذ

عندما يقوم المستخدم بإغلاق أيٍ من النوافذ الوليدة أو إغلاق البرنامج بالكامل، يقوم البرنامج باختبار قيمة الخاصية TextSaved، فإذا كانت False، يتم إظهار رسالة إلى المستخدم تسأله إذا كان يرغب فى حفظ التعديلات أم لا. لأداء ذلك، قم بإدخال الكود التالى داخل حدث إغلاق النافذة الوليدة كما يلى:

```
Private Sub frmChild_Closing(ByVal sender As Object, ByVal e
As System.ComponentModel.CancelEventArgs) Handles
MyBase.Closing
    If Me.TextSaved = False Then
      Dim sMsg As String
```

```
        Dim nResult As Integer
sMsg = "Save changes to "
        If Me.FileName = "" Then
sMsg += "this untitled document"
        Else
sMsg += Mid(sFileName, InStrRev(sFileName, "\") + 1)
        End If
sMsg += " before closing it?"
nResult = MessageBox.Show(sMsg, "One Moment",
MessageBoxButtons.YesNoCancel)
        Select Case nResult
           Case DialogResult.Cancel
e.Cancel = True
           Case DialogResult.Yes
SaveFile()
           Case Else
        End Select
     End If
  End Sub
```

لإغلاق النماذج من خلال القائمة، انقر الخيار **Close** داخل القائمة **File** بالنافذة الوليدة نقراً مزدوجاً ثم قم باستدعاء الوظيفة **()Close** كما يلى:

```
Private Sub mnuFileClose_Click(sender As Object, e As
EventArgs) Handles mnuFileClose.Click
Me.Close()
End Sub
```

طباعة الملفات

إتماماً للفائدة سنقوم بالتعرف على كيفية طباعة محتويات النماذج الوليدة. تابع معنا الخطوات الآتية:

١. قم بإدراج الأداة **PrintDocument** إلى النموذج الوليد وأعد تسميتها إلى **PrntDoc**.

٢. انقر الخيار **Print** داخل قائمة **File** نقراً مزدوجاً ثم قم باستدعاء الوظيفة **()Print** داخل حدث النقر كما يلى:

```
Private Sub mnuFilePrint_Click(sender As Object, e As
EventArgs) Handles mnuFilePrint.Click
```

```
prntDoc.Print()
End Sub
```

٣. قم بإدخال الكود التالى داخل الحدث **PrintPage** الخاص بأداة الطباعة:

```
Private Sub prntDoc_PrintPage(sender As Object, e As
Printing.PrintPageEventArgs) Handles prntDoc.PrintPage
e.Graphics.DrawString(txtMain.Text, New Font("Courier
New", 10, FontStyle.Regular), Brushes.Black, 0, 0)
End Sub
```

لمعرفة المزيد عن الطباعة واستخدام الأداة **PrintDocument**، راجع الجزء الثانى من الكتاب.

وبهذا نكون قد أنشأنا تطبيق متعدد المستندات متكامل يمكنك استخدامه كنواه لتطبيقات أكبر.

إليك الكود الكامل لكلٍ من النافذة الأم والنافذة الوليدة:

أولاً: كود النافذة الأم

```
Imports System.IO
Public Class frmParent
   Private Sub NewChild()
      Dim f As New frmChild()
f.MdiParent = Me
f.FileName = "Untitled "
f.TextSaved = True
f.Show()
   End Sub

   Public Sub New()
MyBase.New()
' This call is required by the designer.
InitializeComponent()

' Add any initialization after the InitializeComponent() call.
NewChild()
   End Sub
```

```
    Private Sub MnuFileExit_Click(sender As Object, e As
EventArgs) Handles MnuFileExit.Click
Me.Close()
    End Sub

    Private Sub MnuFileNew_Click(sender As Object, e As
EventArgs) Handles MnuFileNew.Click
NewChild()
    End Sub

    Private Sub MnuFileOpen_Click(sender As Object, e As
EventArgs) Handles MnuFileOpen.Click
dlgOpen.ShowDialog()
    End Sub
Private Sub dlgOpen_FileOk(sender As Object, e As
System.ComponentModel.CancelEventArgs) Handles
dlgOpen.FileOk
        Dim f As New frmChild()
        Dim nType As Integer = FreeFile()
        Dim myFile As StreamReader
f.MdiParent = Me
myFile = File.OpenText(dlgOpen.FileName)
f.txtMain.Text = myFile.ReadToEnd
f.txtMain.Select(0, 0)
myFile.Close()
f.FileName = dlgOpen.FileName
f.Show()
    End Sub

    Private Sub mnuWindowTileH_Click(sender As Object, e As
EventArgs) Handles mnuWindowTileH.Click
Me.LayoutMdi(MdiLayout.TileHorizontal)
    End Sub

    Private Sub mnuWindowTileV_Click(sender As Object, e As
EventArgs) Handles mnuWindowTileV.Click
Me.LayoutMdi(MdiLayout.TileVertical)
    End Sub
```

```
    Private Sub mnuWindowCascade_Click(sender As Object, e
As EventArgs) Handles mnuWindowCascade.Click
Me.LayoutMdi(MdiLayout.Cascade)
  End Sub
End Class
```

ثانياً كود النافذة الوليدة

```
Imports System.IO
Public Class frmChild
  Public TextSavedAs Boolean
  Private sFileName As String
  Public Property FileName() As String
    Get
      Return sFileName
    End Get
Set(ByVal Value As String)
sFileName = Value
      If Trim(sFileName) = " " Then
Me.Text = "Untitled"
      Else
Me.Text = Mid(sFileName, InStrRev(sFileName, "\") + 1)
      End If
    End Set
  End Property

 Private Sub txtMain_TextChanged(sender As Object, e As
EventArgs) Handles txtMain.TextChanged
Me.TextSaved = False
 End Sub
 Private Sub SaveFile()
    Dim myFile As StreamWriter
    If sFileName = "" Then
SaveFileAs()
    End If
myFile = File.CreateText(sFileName)
myFile.Write(txtMain.Text)
myFile.Close()
Me.TextSaved = True
  End Sub
```

```
   Private Sub SaveFileAs()
dlgSave.FileName = Me.FileName
dlgSave.ShowDialog()
   End Sub

   Private Sub mnuFileSave_Click(sender As Object, e As
EventArgs) Handles mnuFileSave.Click
      If sFileName> "" Then
SaveFile()
      Else
SaveFileAs()
      End If
   End Sub

   Private Sub mnuFileSaveAs_Click(sender As Object, e As
EventArgs) Handles mnuFileSaveAs.Click
SaveFileAs()
   End Sub

   Private Sub dlgSave_FileOk(sender As Object, e As
System.ComponentModel.CancelEventArgs) Handles
dlgSave.FileOk
      If e.Cancel Then
Me.FileName = dlgSave.FileName
SaveFile()
      End If
   End Sub

   Private Sub frmChild_Closing(sender As Object, e As
System.ComponentModel.CancelEventArgs) Handles
MyBase.Closing
      If Me.TextSaved = False Then
         Dim sMsg As String
         Dim nResult As Integer
sMsg = "Save changes to "
         If Me.FileName = "" Then
sMsg += "this untitled document"
         Else
sMsg += Mid(sFileName, InStrRev(sFileName, "\") + 1)
         End If
```

```
sMsg += " before closing it?"
nResult = MessageBox.Show(sMsg, "One Moment",
MessageBoxButtons.YesNoCancel)
          Select Case nResult
             Case DialogResult.Cancel
e.Cancel = True
             Case DialogResult.Yes
SaveFile()
             Case Else
          End Select
       End If
   End Sub

   Private Sub mnuFileClose_Click(sender As Object, e As
EventArgs) Handles mnuFileClose.Click
Me.Close()
   End Sub

   Private Sub mnuFilePrint_Click(sender As Object, e As
EventArgs) Handles mnuFilePrint.Click
prntdoc.Print()
   End Sub

   Private Sub prntdoc_PrintPage(sender As Object, e As
Printing.PrintPageEventArgs) Handles prntdoc.PrintPage
e.Graphics.DrawString(txtMain.Text, New Font("Courier New",
10, FontStyle.Regular), Brushes.Black, 0, 0)
   End Sub
End Class
```

الباب الخامس

أدوات التحكم المتقدمة

٢١ . أدوات التحكم الرئيسية.

٢٢ . أدوات التحكم فى عرض البيانات.

٢٣ . استخدام الحاويات.

٢٤ . المزيد عن أدوات التحكم.

٢٥ . إنشاء أدوات التحكم المخصصة.