الباب الرابع
العمل مع الملفات

١٥ . العمل مع الملفات.

١٦ . المزيد عن الملفات.

الفصل الخامس عشر
العمل مع الملفات

تعتبر الملفات من العناصر الأساسية المستخدمة فى تخزين البيانات. سنقوم فى هذا الفصل بالتعرف على الوظائف والتصنيفات والمسميات المختلفة المستخدمة للتعامل مع الملفات داخل **Visual Basic 2008** على أن نستكمل الحديث فى الفصل القادم إن شاء الله.

بانتهاء هذا الفصل ستتعرف على:

- التعامل مع نظام الملفات من خلال **Visual Basic 2008**.
- التأكد من وجود الملفات والمجلدات.
- سرد الملفات والمجلدات.
- تغيير صفات الملفات والمجلدات.
- عمليات النسخ والحذف وإعادة التسمية.
- العمل مع المسارات.
- مراقبة تغييرات الملفات.
- تشغيل البرامج الأخرى.
- الوصول للملفات باستخدام الكائن **My**.

بغض النظر عن لغة التطوير التى تقوم باستخدامها، ستحتاج فى أى وقت إلى كتابة الكود الذى يقوم بالتفاعل مع ملفات نظام التشغيل. فربما أردت نسخ بعض الملفات أو تشغيل البرامج الأخرى من خلال الملفات الأخرى. يمكنك أداء كل ذلك داخل نظام التشغيل من خلال مستكشف Windows أو نافذة محث الأوامر. كما يمكنك أيضاً استخدام الملفات فى تخزين البيانات واسترجاعها. سنتعرف فى هذا الفصل على كيفية أداء هذه المهام من خلال كود Visual Basic وذلك من خلال تخزين البيانات واسترجاعها من الملفات النصية.

التعامل مع نظام الملفات من خلال Visual Basic 2008

يعبر مستكشف Windows دائماً عن محتويات وحدات التخزين المثبتة على حاسبك بصورة رسومية من خلال تمثيل الملفات والمجلدات برموزها المميزة. ويمكنك من خلال هذا المستكشف إجراء العديد من العمليات المختلفة على هذه الملفات وتلك المجلدات كعمليات النسخ والنقل والحذف وإعادة التسمية ... الخ. يمكنك بلا شك إجراء جميع هذه العمليات من خلال كود Visual Basic 2008 عن طريق استخدام المسمى System.IO الذى يوجد تلقائياً داخل جميع المشروعات التى تقوم بإنشائها، لذا تحتاج فقط إلى تضمينه فى بداية النموذج من خلال العبارة التالية:

```
Imports System.IO
```

وتأتى قوة هذا المسمى من احتوائه على العديد من التصنيفات الهامة المستخدمة فى أداء العمليات المختلفة على الملفات والمجلدات وعلى رأسها تلك الموضحة فى جدول ١٥-١ التالى.

جدول ١٥-١ تصنيفات المسمى System.IO

التصنيف	الوصف
FileSystemInfo	يمثل أى عنصر داخل نظام الملفات سواءً كان ملف أو مجلد. ولا يمكنك إنشاء حالة من كائن هذا التصنيف باستخدام كلمة

التصنيف	الوصف
	New وإنما يمكنك استخدام التصنيف عند معالجة الملفات والمجلدات فى نفس الوقت
FileInfo	يشتق من التصنيف FileSystemInfo ويستخدم فى معالجة ملفات نظام التشغيل حيث يمثل كل كائن FileInfo تقوم بإنشائه أحد الملفات الموجودة لديك
DirectoryInfo	يشتق أيضاً من التصنيف FileSystemInfo ويتيح لك التحكم فى مجلدات نظام الملفات
File	يحتوى على الوظائف الساكنة المستخدمة لأداء العمليات المختلفة على ملفات نظام التشغيل
Directory	يحتوى على الوظائف الساكنة المستخدمة لأداء العمليات المختلفة على مجلدات نظام التشغيل

إذا دققت النظر فى الجدول السابق، تلاحظ تكرار الوظائف داخل بعض التصنيفات، فكل من التصنيف File والتصنيف FileInfo على سبيل المثال يحتوى على الوظائف التى تحدد وجود الملف أو حذفه. والسبب فى ذلك أن التصنيف File فى حالة الملفات والتصنيف Directory فى حالة المجلدات يحتويان على جميع الوظائف الساكنة Static Methods بينما يحتوى التصنيفان FileInfo وDirectoryInfo على وظائف الحالة Instance Methods. والوظيفة الساكنة كما تعلم هى الوظيفة التى يمكنك استدعائها من خلال التصنيف نفسه وليس كائن من كائنات هذا التصنيف، والعكس صحيح بالنسبة لوظيفة الحالة. وعلى ذلك فعند العمل مع الملفات، يفضل استخدام الوظائف الساكنة إذا لم ترغب فى إجراء العديد من العمليات على نفس الملف أو المجلد.

التأكد من وجود الملفات والمجلدات

إذا حاولت الوصول إلى قاعدة بيانات أو فتح أحد الملفات الغير موجودة، سيقوم المترجم بالاعتراض وإظهار رسالة الخطأ المناسبة. لذا يمكنك استخدام الوظيفة **File.Exists()** قبل فتح الملف للتأكد من وجوده بالفعل كما فى الكود التالى:

```
Dim sFileToCheck As String = "c:\Books.doc"
If Not File.Exists(sFileToCheck) Then
    MessageBox.Show("Cannot find the file. Please try again!")
    Application.Exit()
End If
```

حيث تحتوى الوظيفة **Exists()** على معامل واحد عبارة عن مسار الملف الذى ترغب فى البحث عنه وتقوم بإرجاع القيمة **True** فى حالة وجود الملف أو القيمة **False** فى حالة العكس. وهناك بعض الملاحظات أثناء استخدامك للوظيفة **Exists()** وهى:

- لا تهتم الوظيفة **Exists()** بحالة حروف اسم الملف.
- يجب أن تقوم بتحديد مسار الملف أثناء استدعاء الوظيفة بغض النظر عن وجود الملف فى المسار الحالى أم لا.
- يمكنك تحديد المسار كاملاً أو بالنسبة للمسار الحالى.
- لا يمكنك استخدام الحروف العامة ? و* داخل الوظيفة **Exists()**.
- يمكنك استخدام مسارات الخادم مثل **\\server\share\directory\filename** مع الوظيفة **Exists()**.

يمكنك استخدام الوظيفة **Directory.Exists()** للتأكد من وجود المجلدات بنفس طريقة استخدام الوظيفة **File.Exists()**.

يحتوى التصنيفان **FileInfo** و**DirectoryInfo** أيضاً على الخاصية **Exists** التى يمكنك استخدامها مع أحد الكائنات المعرفة مسبقاً بنفس طريقة استخدام الوظيفة **Exists()** فى التصنيفين **File** و**Directory**.

سرد الملفات والمجلدات

تعتبر القدرة على سرد الملفات والمجلدات الموجودة بدليل معين من العمليات الهامة التى كثيراً ما تستخدم داخل نظام الملفات وذلك من خلال الوظيفة **()GetFiles** التى تقوم بإرجاع قائمة من الملفات. وبمجرد الحصول على هذه القائمة، يمكنك عرضها للمستخدم على الشاشة أو تخزينها داخل قاعدة بيانات أو تنفيذ العديد من المهام المختلفة عليها. فمثلاً يوضح الكود التالى استخدام الوظيفة **()Directory.GetFiles** لملء مربع سرد بأسماء الملفات التنفيذية داخل المجلد **C:\Windows**:

```
Dim sFileList() As String
sFileList = Directory.GetFiles("C:\Windows","*.exe")
ListBox1.Items.AddRange(sFileList)
```

حيث تقوم الوظيفة **()GetFiles** بالبحث داخل المجلد المحدد عن الملفات وإرجاعها فى صورة مصفوفة من النصوص تحتوى على المسار الكامل لكل ملف. ولكى تقوم بتحديد المجلد محل البحث، يجب أن تقوم بتعيين معامل واحد على الأقل وهو اسم المجلد نفسه، إلى جانب معامل اختيارى يعبر عن نوع الملفات التى ترغب فى البحث عنها.

وبالإضافة إلى الوظيفة **()GetFiles**، يحتوى التصنيف **Directory** أيضاً على بعض الوظائف المشابهة مثل **()GetDirectories** و**()GetFileSystemEntries** التى يمكنك استخدامها للحصول على مصفوفة من المجلدات الفرعية أو الملفات والملفات الفرعية على الترتيب. وعلى الرغم من كفاءة هذه الوظائف وسهولة استخدامها إلا أنها لا تتناسب مع العمليات الأكثر تعقيداً. فعلى سبيل المثال، ربما أردت معالجة كل ملف ككائن مستقل وفى هذه الحالة يمكنك استخدام خصائصه المختلفة كالحجم والامتداد لتعيين الملفات التى يتم إجراء عملياتك عليها وفى هذه الحالة يمكنك استخدام بعض الوظائف الأخرى مثل الوظيفة **()GetFileSystemInfos** كما فى الكود التالى:

```
Dim dirTemp As DirectoryInfo
Dim fsiTemp As FileSystemInfo

dirTemp = New DirectoryInfo("C:\")
```

```
lstFiles.DisplayMember = "Name"
For Each fsiTemp In dirTemp.GetFileSystemInfos()
        lstFiles.Items.Add(fsiTemp)
Next
```

قمنا فى هذا الكود بدايةً بإنشاء حالة من التصنيف DirectoryInfo والتى تمثل الـدليل C:\ وبعد ذلك قمنا باستدعاء الوظيفة GetFileSystemInfos() التى تقوم بإرجـاع مجموعة من الكائنات المصاحبة لكل عنصر موجود داخل المجلد C:\ وأخيراً يتم إضافة جميع الكائنات FileSystemInfo إلى مجموعة عناصر مربع السرد.

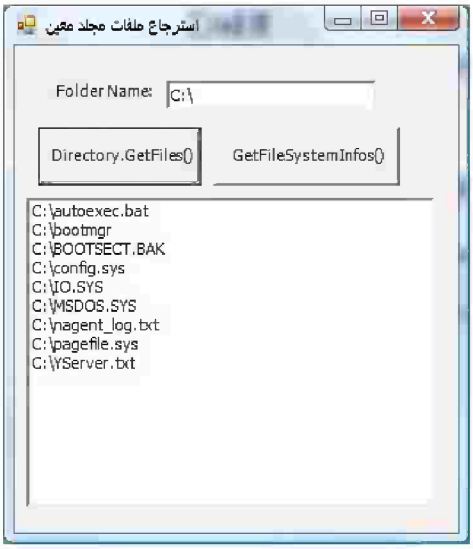
إذا أردت العمل مع الملفات فقط أو المجلدات فقط وليس كلاهما، اسـتخدم الوظيفة GetFiles() أو الوظيفة GetDirectories() بدلاً مـن الوظيفـة العامة GetFileSystemInfos(). كما تحتوى جميع الوظائف السابقة على إمكانية تحديد معامل شكل البحث الذى يمكنك من خلاله تحديد أنواع الملفات التى ترغب فى تضمينها كالملفات التنفيذية مثلاً.

لمزيد من التعرف على كيفية سرد الملفات والمجلدات وطريقة التعامل معها، دعنـا نـرى الكود التالى والذى يتم فيه استرجاع خصائص الكائن عن طريق نقر اسم الملـف داخـل مربع السرد نقراً مزدوجاً (يوجد هذا الكود داخل المشروع ListFiles علـى القـرص المدمج المرفق بالكتاب):

```
Private Sub lstFiles_DoubleClick(ByVal sender As Object, ByVal
e As System.EventArgs) Handles lstFiles.DoubleClick

  Dim fsiTemp As FileSystemInfo
  Dim strMessage As String

  fsiTemp = CType(lstFiles.SelectedItem, FileSystemInfo)
  If lstFiles.SelectedItem.GetType Is GetType(DirectoryInfo) Then
    Dim dirTemp As DirectoryInfo
    dirTemp = CType(fsiTemp, DirectoryInfo)
    strMessage = "You just clicked on a directory." & vbCrLf
    strMessage &= "The directory's parent is " &
                          dirTemp.Parent.ToString & vbCrLf
```

```
ElseIf lstFiles.SelectedItem.GetType Is GetType(FileInfo) Then
    Dim filTemp As FileInfo
    filTemp = CType(fsiTemp, FileInfo)
    strMessage = "You just clicked on a file." & vbCrLf
    strMessage &= "The size of the file is " &
                          filTemp.Length.ToString & vbCrLf
    strMessage &= "The complete path to it is " &
                                          filTemp.FullName
End If
MessageBox.Show(strMessage)

End Sub
```

يوضح هذا الكود كيفية استخدام خصائص التصنيفات **DirectoryInfo** و**FileInfo** والتصنيف العام **FileSystemInfo**. ولكن يجب أن تتذكر دائماً، أنك إذا لم تكن بحاجة إلى الوصول إلى الكائن بالكامل، فمن الأفضل تخزين الملفات داخل مربع سرد كنصوص وليس كائنات (انظر شكل ١٥-١).



شكل ١٥-١ استرجاع الملفات الموجودة بأحد المجلدات

يوضح الكود التالى كيفية الوصول إلى المجلد **C:\Program Files** وجميع مجلداته الفرعية ثم إضافة جميع الملفات ذات الامتداد **JPG** التى تصادفه. ومن ثمَّ يستطيع

المستخدم بعد ذلك نقر أى ملف لوضعه كخلفية للنموذج (يوجد هـذا الكـود داخـل المشروع FindPics على القرص المدمج المرفق بالكتاب، ويحتوى نموذج المشروع علـى مربع سرد وزر أمر فقط):

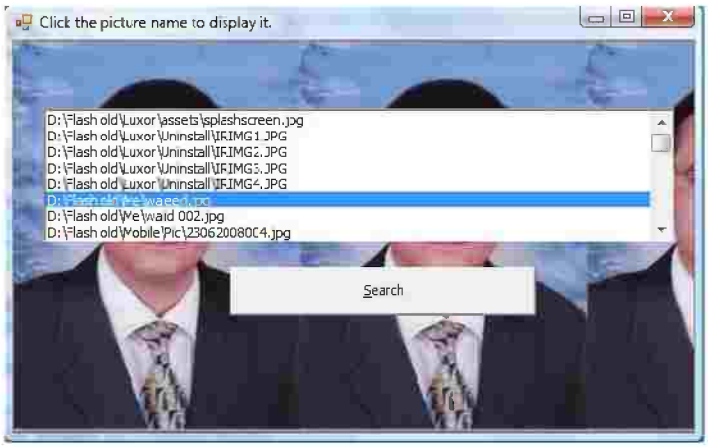
```
1.  Private Sub FindPictures(ByVal fsiTemp As
    FileSystemInfo)
2.  'يتم تمرير ملف أو مجلد إلى هذه الدالة
3.  'فإذا كان الملف عبارة عن صورة JPG
4.  'يتم على الفور إضافتها إلى مربع السرد
5.  'أما إذا كانت مجلداً، فسيتم نداء الدالة داخل هذا المجلد

6.  If fsiTemp.GetType Is GetType(FileInfo) Then
7.      If fsiTemp.Extension.ToUpper = ".JPG" Then
8.          lstFiles.Items.Add(fsiTemp.FullName)
9.      End If
10. Else
11.     Dim dirTemp As New DirectoryInfo(fsiTemp.FullName)
12.     Dim fsiNew As FileSystemInfo
13.     Me.Text = "Searching " & dirTemp.Name
14.     For Each fsiNew In dirTemp.GetFileSystemInfos
15.         Call FindPictures(fsiNew)
16.     Next
17. End If
18. End Sub

19. Private Sub lstFiles_SelectedIndexChanged(ByVal sender
    As System.Object, ByVal e As System.EventArgs) Handles
    lstFiles.SelectedIndexChanged
20. If Not (lstFiles.SelectedItem Is Nothing) Then
21.   Me.BackgroundImage =
22.             Image.FromFile(lstFiles.SelectedItem.ToString)
23. End If
24. End Sub
```

```
25.  Private Sub btnSearch_Click(ByVal sender As
     System.Object, ByVal e As System.EventArgs) Handles
     btnSearch.Click
26.    Dim dirTemp As New DirectoryInfo("D:")
27.    lstFiles.Items.Clear()
28.    Call FindPictures(dirTemp)
29.    dirTemp = Nothing
30.    Me.Text = "Click the picture name to display it."
31.  End Sub
```

وعن هذا الكود، نوضح ما يلى:

- تحتوى السطور من ١ إلى ١٨ على كود الإجـراء الفرعـى **FindPictures()** المستخدم للبحث عن الصور وتضمينها داخل مربع السرد. ويتم فى هذا الإجـراء اختبار الكائن الممرر الذى قد يكون ملفاً أو مجلداً. فإذا كان الكائن عبـارة عـن ملف صورة بالامتداد **JPG**.، يتم إضافته إلى مربع السرد، أما إذا كان مجلداً، فيتم استدعاء الإجراء مرةً أخرى لتنفيذه على هذا المجلد، حيث يطلق على هذه العملية "التكرار" **Recursion** وهو نداء الإجراء لنفسه. كما يتم تغيير عنوان النموذج ليعكس مكان البحث الحالى داخل المجلد.
- تحتـوى السـطور مـن ١٩ إلى ٢٤ علـى كـود الإجـراء **lstFiles_SelectedIndexChanged()** الذى يتم استدعاءه بمجرد اختيـار المستخدم لعنصر آخر داخل مربع السرد، حيث يقوم هذا الإجراء بتعـيين اسـم ملـف الصـورة المختـار كخلفيـة للنمـوذج مـن خـلال الخاصـية **BackGroundImage**.
- تحتوى السطور من ٢٥ إلى ٣١ على كود إجراء حدث نقر الـزر **btnSearch** المستخدم لإجراء عملية البحث داخل القرص **D:**.

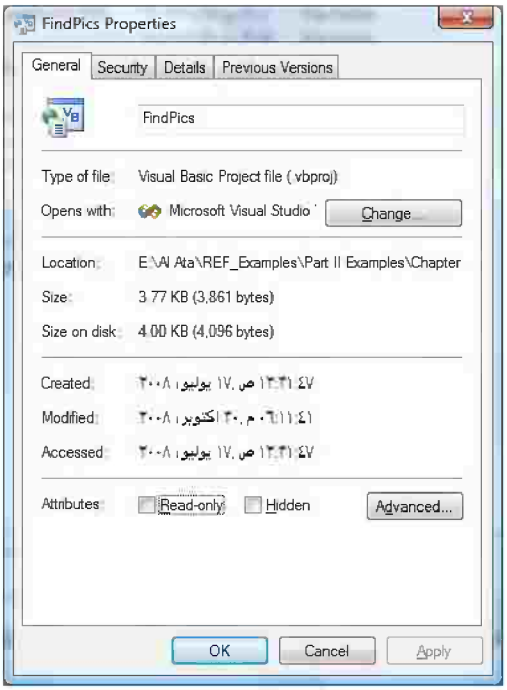
قم الآن بتشغيل الكود ثم انقر زر البحث، تلاحظ ملء مربع السرد بجميع ملفات الصور ذات الامتداد **JPG**. داخل القرص **D:** (يمكنك تعيين مسار آخر إن أحببت). انقر أياً من هذه الصور، تلاحظ ظهور هذه الصورة كخلفية للنموذج (انظر شكل ١٥-٢).



شكل ١٥-٢ عرض الصورة المختارة كخلفية للنموذج

تغيير صفات الملفات والمجلدات

يمكنك قراءة المعلومات المصاحبة للملف أو المجلد أو حتى تعديلها عن طريق خصائص ووظائف كائن التصنيف **FileSystemInfo**. فإذا قمت على سبيل المثال بنقر أحد الملفات أو المجلدات داخل نافذة مستكشف **Windows** بزر الفأرة الأيمن ثم اخترت **Properties** من القائمة الموضعية، سيظهر مربع حوارى شبيه بذلك الموضح فى شكل ١٥-٣ التالى.



شكل ١٥-٣ مربع خصائص الملف كما يظهر داخل Windows

يوضح الكود التالى كيفية تغيير خاصية القراءة فقط Read-Only المصاحبة لأحد الملفات والتى تنعكس بالطبع على مربع الخصائص داخل مستكشف Windows.

```
Dim MyFile As FileInfo
MyFile = New FileInfo("c:\myfile.txt")
If MyFile.Attribures And IO.FileAttributes.ReadOnly Then
        MessageBox.Show("Your file is already set to read-only
                                                        mode!")
Else
        MessageBox.Show("The read-only flag has been set!")
        MyFile.Attributes = MyFile.Attributes or
        IO.FileAttributes.ReadOnly
End If
```

وهناك العديد من الصفات المصاحبة للملفات والمجلدات مثـــل Read-Only وHidden وArchive، حيث يتم تخزين قيم هذه الصفات داخل الخاصية Attributes فى صـــورة

Bits لذا يجب أن تقوم باستخدام العمليات المنطقية على مستوى البت إذا أردت تعيين هذه القيم أو تغييرها كما فعلنا فى الكود السابق من خلال استخدام المعامل AND والمعامل OR.

عمليات النسخ والحذف وإعادة التسمية

يحتوى التصنيف File على الوظائف التى تمكنك من نسخ الملفات أو حذفها أو حتى إعادة تسميتها. فعلى سبيل المثال، يمكنك استخدام الوظيفة File.Copy() لإنشاء نسخة احتياطية من الملف هكذا:

```
File.Copy("C:\myfile.txt", "C:\myfileCopy.txt", True)
```

حيث تحتوى هذه الوظيفة على المعاملات التالية:

- الملف الأصلى Source File وهو الملف الذى ترغب فى نسخه. فإذا لم يتم العثور على هذا الملف، يتم على الفور طرح الاستثناء المناسب.
- الملف الهدف Destination File وهو المسار المستخدم لتعيين اسم ومكان الملف المنسوخ (يمكنك استخدام نفس الاسم الأصلى للملف إذا قمت بوضعه داخل مجلد آخر).
- الكتابة فوق الملف الموجود Overwrite Existing File وهذا المعامل اختيارى ويوضح إذا ما كنت ترغب فى الكتابة فوق الملف الموجود مسبقاً أم لا وذلك باستخدام القيمة True. أما إذا قمت بإهمال هذا المعامل أو تخصيصه بالقيمة False ثم عُثر على ملف بنفس الاسم فى مكان عملية النسخ، فيتم طرح استثناء (خطأ) على الفور، لذا يجب أن تقوم بمعالجة هذه الحالة من خلال الكود.

وكما هو الحال مع الوظائف المشتركة، توجد وظيفة مصاحبة للاستخدام مع حالة الكائن. فإذا قمت بإنشاء كائن FileInfo، يمكنك استخدام الوظيفة CopyTo() وذلك بتعيين المعامل الثانى والثالث فقط فى الوظيفة السابقة لأن المعامل الأول معروف مسبقاً وهو الملف المرتبط بالكائن FileInfo كما فى الكود التالى:

```
Dim fileTemp As New File("C:\myfile.txt")
```

```
fileTemp.CopyTo("C:\myfileCopy.txt",False)
```

تتسبب الوظيفتان Copy() وCopyTo() فى طرح استثناء (خطأ بالبرنامج) إذا حاولت الكتابة فوق أحد الملفات التى تحتوى على الخاصية Read-Only حتى وإذا احتوى المعامل الأخير على القيمة True. فإذا أردت الكتابة فوق ملف للقراءة فقط، قم أولاً بتعطيل هذه الخاصية كما أوضحنا منذ قليل.

لا يحتوى التصنيف DirectoryInfo على الوظيفة Copy() أو الوظيفة CopyTo() وإنما يمكنك إنشاء مجلد جديد باستخدام الوظيفة Create() ثم نسخ الملفات بعد ذلك إلى هذا المجلد. أما عمليات الحذف وإعادة التسمية فهى واحدة بالنسبة للملفات والمجلدات، حيث يمكنك استخدام الوظائف التالية:

- الوظيفة Move() وهى عبارة عن وظيفة ساكنة تستخدم لإعادة تسمية الملف أو المجلد وتحتوى على معاملين فقط هما المصدر Source() والهدف Destination().
- الوظيفة MoveTo() وهى وظيفة حالة تستخدم فى إعادة تسمية ملف أو مجلد.
- الوظيفة Delete() وتقوم بحذف الملف أو المجلد المحدد ولا يمكنك معها استخدام الحرفين الشاملين (? ، *). وعند استخدام هذه الوظيفة، يجب التأكد أولاً من أن الملف ليس للقراءة فقط وإلا سيتم طرح الاستثناء المناسب.

العمل مع المسارات

كى تتمكن من التعامل مع أحد الملفات من خلال Visual Basic، يجب أن تقوم أولاً بتحديد مسار هذا الملف حيث يعبر المصطلحان Filename وPathname عن نفس المعنى. أما كلمة Path فتعنى تضمين بعض معلومات المجلد مع اسم الملف. يوضح الكود التالى ثلاثة أمثلة للمسارات الصحيحة للملفات:

- C:\MyApplication\Sounds\ding.wav
- Sounds\ding.wav
- ding.wav

فعلى فرض تثبيت التطبيق داخل المجلد C:\MyApplication ونحتاج إلى الوصول لبعض ملفات الصوت الموجودة بالمجلد الفرعى C:\MyApplication\Sounds، فإن المسارات الثلاثة السابقة تشير إلى نفس الملف ding.wav، حيث يعبر السطر الأول عن المسار المطلق Absolute Path بينما يعبر السطران التاليان عن مسارات نسبية Relative Paths.

البحث عن الملفات من خلال المسارات النسبية

تبنى المسارات النسبية دائماً على المجلد الحالى الذى يمكنك تخيله كما لو كان المجلد المفتوح حالياً داخل مستكشف Windows أو داخل نافذة محث الأوامر. وحينما يحتاج التطبيق إلى إيجاد الملفات الأخرى من خلال المسارات النسبية، يمكنك استخدام الخاصيتين التاليتين:

- الخاصية Directory.GetCurrentDirectory الموجودة داخل المسمى System.IO والمستخدمة فى تعيين أو استرجاع الدليل (المجلد) الحالى.
- الخاصية Application.StartupPath الموجودة داخل المسمى System.Windows.Forms والمستخدمة فى استرجاع المجلد الذى تم من خلاله تشغيل التطبيق.

وحينما تقوم بتشغيل أحد التطبيقات النوافذية، يمكنك استخدام الوظيفة GetCurrentDirectory() والخاصية StartupPath() للحصول على مجلد التطبيق الذى يحتوى بدوره على الملف التنفيذى لهذا التطبيق. ومع تشغيل التطبيق، ربما يتغير المجلد الحالى إلَّا أن مسار بداية التطبيق الممثل فى الخاصية StartupPath يظل كما هو. لنفترض مثلاً أن تطبيقك فى المثال السابق يحتاج إلى أداء العديد من المهام داخل المجلد Sounds، يمكنك فى هذه الحالة جعل هذا المجلد هو المجلد الحالى من خلال الوظيفة SetCurrentDirectoy() كما فى الكود التالى:

```
Directory.SetCurrentDirectory(Application.StartupPath &
"\Sounds")
```

يتم استخدام الشرطة المائلة (\) لفصل أسماء المجلدات داخل مسار الملف، لذا قمنا فى هذا الكود باستخدام هذه الشرطة قبل اسم المجلد Sounds. وبدءاً من Visual Basic.NET تم تصميم التصنيف Path لتسهيل العمل مع مسارات الملفات. فعلى سبيل المثال، يمكنك استخدام الوظيفة ()Combine التى تنتمى للتصنيف Path فى تعيين مسار الملف، حيث تقوم هذه الوظيفة بإضافة الشرطة المائلة تلقائياً كما فى الكود التالى:

```
strPath = Directory.Combine("C:\", "Program Files")
strPath = Directory.Combine(strPath, "Netmeeting")
strPath = Directory.Combine(strPath, "ding.wav")
```

وبمجرد تنفيذ هذا الكود، سيحتوى المتغير النصى strPath على المسار التالى:

```
C:\Program Files\Netmeeting\ding.wav
```

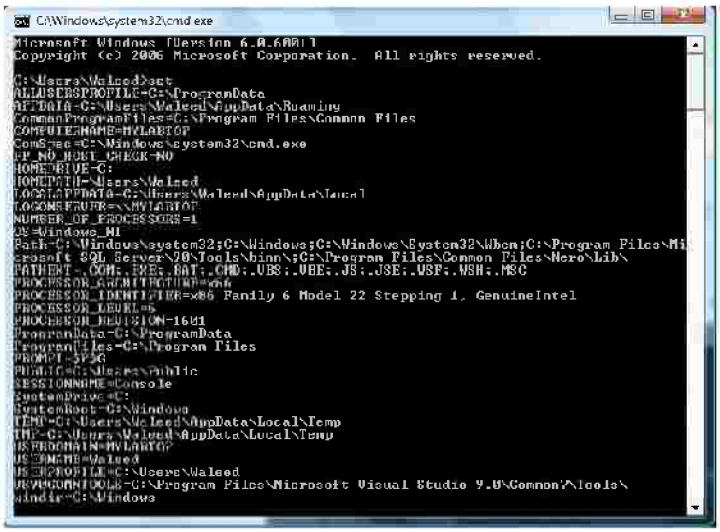
ولكن ضع فى خلدك أن الوظيفة ()Combine تقوم بتجميع المسارات كنصوص فقط، إلا أنها لا تقوم بالتأكد من صحة هذه المسارات.

البحث عن المجلدات الخاصة

من المجلدات العديدة الموجودة على قرصك الصلب، هناك بعض المجلدات المستخدمة من قِبَل نظام التشغيل والتى تقوم ببعض الأغراض الخاصة مثل المجلد My Documents الذى يتيح للمستخدمين وضع جميع المستندات فى مكان واحد، والمجلد Temp المستخدم فى تخزين الملفات المؤقتة Temporary Files، هذا بالإضافة إلى مجلد Windows نفسه.

تحتوى هذه المجلدات على مسارات مختلفة تبعاً لنظام التشغيل المستخدم. فقد اعتاد مستخدمو Windows 95 على تثبيت نظام التشغيل داخل المجلد C:\Windows، بينما استخدم نظام التشغيل Windows 2000 و Windows NT المجلد C:\WINNT، كما يتم تخصيص المجلدات تبعاً للمستخدم الذى قام بالدخول إلى نظام التشغيل كما فى Windows 2000 وWindows XP وWindows Vista، حيث يتم وضع المجلدات My Documents وTemp الخاصة بكل مستخدم على حده داخل المجلد Documents And Settings.

إذا أردت الوصول إلى هذه المجلدات، يمكنك استخدام متغيرات بيئة نظام التشغيل. وللتعرف على هذه المتغيرات، افتح نافذة محث الأوامر ثم قم بتنفيذ الأمر SET، تحصل على قائمة بمسارات المجلدات الخاصة (انظر شكل ١٥-٤).



شكل ١٥-٤ الحصول على مسارات المجلدات الخاصة

أما إذا أردت الوصول إلى هذه المتغيرات من خلال الكود، فيمكنك استخدام التصنيف Environment الذى يحتوى على العديد من الوظائف والخصائص المستخدمة للتعامل مع هذه المتغيرات والتى على رأسها تلك الموضحة فى جدول ١٥-٢ التالى.

جدول ١٥-٢ خصائص ووظائف التعرف على متغيرات بيئة نظام التشغيل

الاسم	الوصف
GetEnvironmentVariable()	تستخدم هذه الوظيفة لإرجاع قيمة أحد المتغيرات
SystemDirectory	تستخدم هذه الخاصية لإرجاع مسار المجلد System الخاص بنظام التشغيل

OSVersion	تستخدم هذه الخاصية للتعرف على إصـــدار نظام التشغيل المثبت على الحاسب

فعلى سبيل المثال، يقوم الكود التالى بإرجاع مسار المجلد My Documents:

```
strPath = Environment.GetEnvironmentVariable
("USERPROFILE")
strPath = Path.Combine(strPath, "My Documents")
```

مراقبة تغييرات الملفات

أحياناً ترغب فى التعرف على الملفات الناتجة من خلال شخص أو تطبيق آخر. فمثلاً ربمـــا أردت الحصول على ملف Excel موجود على الويب كلما تم تحديث هـــذا الملـــف. أو حينما يقوم أحد الأشخاص بإرسال ملف بيانات من خلال البريد الإلكترونى، وفى هـــذه الحالة تحتاج إلى إحضار هذا الملف إلى قاعدة البيانات دورياً كى تحصل علـــى البيانـــات الجديدة حيث يتم ذلك بمساعدة الوظيفة Find.Exists(). لكن من حسن الحظ احتواء نطاق .NET. على التصنيف FileSystemWatcher الذى يقـــوم بـــإعلام تطبيقـــك بالأنواع المختلفة من التغييرات التى تتم على الملفات.

سنقوم فيما يلى بإنشاء تطبيق لمراقبة أحد المجلدات لإضافة ملفات جديدة أو تعديل الملفات الموجودة من خلال التصنيف FileSystemWatcher، حيث يتم إظهار هذه التغييرات داخل مربع سرد. لأداء ذلك، تابع معنا الخطوات الآتية (يوجد هذا المشروع على القرص المدمج المرفق بالكتاب باسم Watching):

١. قم بإنشاء مشروع نوافذى جديد باسم مناسب وليكن Watching.

٢. قم بإضافة الكود التالى إلى تصنيف النموذج الافتراضى:

```
1.  Private WithEvents fsWatchObj As FileSystemWatcher

2.  Private Sub SetupWatcher(ByVal PathToWatch As String)
3.   fsWatchObj = New FileSystemWatcher(PathToWatch)
4.   fsWatchObj.IncludeSubdirectories = False
5.   fsWatchObj.EnableRaisingEvents = True
```

```
6.    AddHandler fsWatchObj.Created, AddressOf FileChanged
7.    AddHandler fsWatchObj.Changed, AddressOf FileChanged
8.    AddHandler fsWatchObj.Deleted, AddressOf FileChanged
9.    AddHandler fsWatchObj.Renamed, AddressOf
      FileRenamed

10.   End Sub

11.   Private Sub FileChanged(ByVal s As System.Object, ByVal
      EventInfo As FileSystemEventArgs)
12.    Dim DisplayMessage As String
13.    DisplayMessage = "File " & EventInfo.Name & " was "
14.    DisplayMessage &= EventInfo.ChangeType.ToString & "
15.    at " & DateTime.Now.ToString
16.    lstFiles.Items.Add(DisplayMessage)
17.   End Sub

18.   Private Sub FileRenamed(ByVal s As System.Object, ByVal
      EventInfo As RenamedEventArgs)
19.    Dim DisplayMessage As String
20.    DisplayMessage = "File" & EventInfo.OldName
21.    DisplayMessage &= " was renamed to " &
22.    EventInfo.Name & " at " & DateTime.Now.ToString
23.    lstFiles.Items.Add(DisplayMessage)
24.   End Sub
```

٣. قم بإضافة مربع سرد إلى النموذج ثم قم بتغيير اسمه إلى **lstFiles**.

٤. قم بإضافة زر أمر من مربع الأدوات إلى النموذج مع تغيير اسمه إلى **btnStart**.

٥. انقر زر **btnStart** نقراً مزدوجاً ثم قم باستدعاء الإجراء **SetupWatcher** الذى أنشأناه منذ قليل كما يلى:

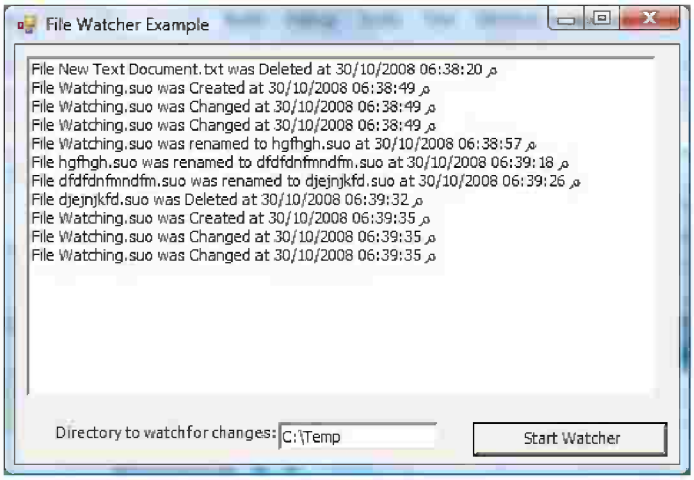
```
Call SetupWatcher("c:\temp")
```

٦. افتح نافذة مستكشف **Windows** وتأكد من وجود المجلد **C:\Temp** وإلا قم بإنشائه.

٧. قم بتشغيل البرنامج ثم انقر زر **btnStart**.

٨. قم بترتيب سطح المكتب حتى تتمكن من مشاهدة نافذة المستكشف ونافذة التطبيق فى نفس الوقت.

٩. قم بنسخ الملفات إلى المجلد **C:\Temp** أو حذفها أو إعادة تسميتها ولاحظ التغييرات داخل مربع السرد الموجود بنموذج التطبيق (انظر شكل ١٥-٥).



شكل ١٥-٥ مراقبة تغييرات الملفات داخل المجلد **C:\Temp**

والآن دعنا نتعرف على محتويات الكود السابق:

- يحتوى السطر رقم ١ على عبارة الإعلان عن الكائن **fsWatchObj** كأحد حالات التصنيف **FileSystemWatcher**.
- تحتوى السطور من ٢ إلى ١٠ على كود الإجراء **()SetupWatcher** والذى يتم من خلاله تعريف الكائن **fsWatchObj** الذى يحتوى على المسار المطلوب مراقبته
- فى السطر رقم ٤ يتم تخصيص القيمة **False** للخاصية **IncludeSubDirectories** حتى لا يتم تضمين المجلدات الفرعية فى عملية المراقبة.

- فى السطر رقم ٥ يتم تخصيص القيمة True للخاصية EnableRaisingEvents حتى يتم تنشيط عملية الاستجابة للأحداث المختلفة.
- فى السطور من ٦ إلى ٩ يتم تعريف إجراءى معالجة أحداث تغيير الملفات داخل المسار المحدد ونعنى بذلك الإجراء FileChanged() والإجراء FileRenamed().
- تحتوى السطور من ١١ إلى ١٧ على كود الإجراء FileChanged() والذى يتم فيه إضافة وصف لعملية التغيير التى تمت على الملف إلى مربع السرد الموجود بالنموذج.
- تحتوى السطور من ١٤ إلى ٢٨ على الإجراء FileRenamed() الذى يتشابه كثيراً مع الإجراء السابق ما عدا كتابة الاسم القديم للملف وكذلك الاسم الناتج عن إعادة التسمية.
- يمكنك استخدام الخاصية Filter المصاحبة للتصنيف FileSystemWater للتحكم فى نوعية الملفات التى ترغب فى مراقبتها داخل المسار المحدد.
- من الممكن استدعاء الحدث FileChanged عدة مرات بالصف الواحد لاحتمال تغيير أكثر من خاصية من صفات الملف مثل الحجم وآخر وقت للكتابة وغيرها أثناء عملية معينة. يمكنك التحكم فى نوع عملية المراقبة من خلال الخاصية NotifyFilter.

تشغيل البرامج الأخرى

ربما أردت من وقتٍ إلى آخر تشغيل أحد برامج Windows من خلال كود Visual Basic. فمن الممكن على سبيل المثال وضع زر داخل النموذج واستخدامه فى تشغيل برنامج الحاسبة ممثلةً فى الملف Calc.exe، كما يمكنك أيضاً اكتشاف أسماء البرامج التى تعمل حالياً أو إيقاف البرامج التى لا تعمل بشكل صحيح.

يمكنك حقيقةً أداء جميع هذه العمليات من خلال التصنيف Process الذى ينتمى للمسمى System.Diagnostic الذى يحتوى بدوره على الوظائف التى تحتاج إلى استدعاءات API فى الإصدارات السابقة من Visual Basic.

تشغيل برامج Windows

أحد الأسئلة المعتادة التى يسألها مستخدمو Visual Basic هو كيفية تشغيل أحد التطبيقات من داخل كود Visual Basic والانتظار حتى ينتهى هذا التطبيق من عمله. يوضح الكود التالى مدى سهولة الكود المستخدم لإنجاز هذه العملية:

```
Dim procNotepad As Process
    procNotepad = Process.Start("notepad.exe")
    procNotepad.WaitForExit(5000)
    If procNotepad.HasExited Then
       MessageBox.Show("Notepad has been closed.")
    Else
       procNotepad.Kill()
       MessageBox.Show("Notepad was still running, so I killed
                                                    it!")
    End If
```

فى هذا الكود يتم تشغيل برنامج NotePad.exe ثم الانتظار لمدة ٥ ثوانى (٥٠٠٠ مللى ثانية) فإذا قام المستخدم بإغلاق البرنامج قبل انقضاء هذه المدة، يتم إظهار رسالة بإغلاق البرنامج، وإلا يتم حذف الملف تلقائياً وإظهار رسالة بذلك.

يمكنك عدم تحديد وقت معين لإنهاء البرنامج بحذف القيمة الموجودة بالوظيفة WaitForExit().

يوضح الكود السابق كيفية تشغيل برنامج Windows والانتظار حتى ينتهى. أما إذا أردت الانتظار حتى ينتهى البرنامج من عمله على أن يتم إعلامك بهذا، فيمكنك إنشاء إجراء احتواء الحدث Process.Exited كما يلى:

```
Dim procNotepad As Process
procNotepad = Process.Start("notepad.exe")
procNotepad.EnableRaisingEvents = True
AddHandler procNotepad.Exited, AddressOf NotepadExitEvent
```

```
Private Sub NotepadExitEvent(ByVal sender As Object, ByVal e
As EventArgs)
    MessageBox.Show("Notepad has just exited!")
End Sub
```

يتيح هذا الكود للمستخدم تشغيل برنامج NotePade. وبمجرد إغلاقه، يتم تشغيل الحدث Exited ومن ثمَّ تشغيل الإجراء NotepadExitEvent الذى يقوم بدوره بإظهار رسالة بإغلاق البرنامج. وحتى يتم تنشيط الحدث Exited، يجب أن تقوم بتخصيص القيمة True للخاصية EnableRizingEvents.

التعرف على البرامج النشطة

يمكنك استخدام التصنيف Process لأداء العديد من المهام التى يصعب علينا حصرها فى هذه السطور البسيطة. فقد رأينا منذ قليل كيفية إغلاق البرامج من خلال الوظيفة Kill() كما يمكنك تغيير أولوية تنفيذ البرنامج وكذلك عرض إحصائيات استخدام الذاكرة والمعالج. ولكن قبل أن تقوم بتغيير خصائص إحدى العمليات، يجب أن تعرف أولاً العملية المصاحبة للكائن Process، وهذا الأمر كما رأينا غاية فى السهولة حينما تقوم بتشغيل البرنامج من خلال كود Visual Basic. لكن ماذا عن البرامج التى تعمل بالفعل خارج هذا الكود ؟. يحتوى التصنيف Process على العديد من الوظائف التى تمكنك من التعامل مع هذه البرامج وعلى رأسها ما يلى:

- الوظيفة GetCurrentProcess() وتقوم بإرجاع كائن التصنيف Process الخاص بالعملية الحالية أو البرنامج الحالى.
- الوظيفة GetProcessByID() وتقوم بإرجاع كائن التصنيف Proccess للعملية (البرنامج) المحددة بالرقم ID وذلك طبقاً للعمود ID داخل نافذة مدير المهام Windows Task Manager.
- الوظيفة GetProcesses() وتقوم بإرجاع مجموعة من كائنات التصنيف Process التى يمثل كل منها عملية من العمليات التى تعمل بالنظام.

- الوظيفة GetProcessesByName() وتقوم بإرجاع مجموعة من كائنات التصنيف Process التى تحتوى على اسم معين.

فعلى سبيل المثال، يقوم الكود التالى بإضافة أسماء عمليات النظام المختلفة إلى مربع السرد lstProcesses من خلال الوظيفة GetProcesses():

```
Dim procTemp As Process
For Each procTemp in Process.GetProcesses()
        lstProcesses.Items.Add(procTemp.ProcessName)
Next
```

بعد الانتهاء من تنفيذ أى عملية (برنامج)، يتم الاحتفاظ بموارد هذه العملية لبعض الوقت، لذا ربما تقوم الوظائف السابقة بإرجاع عمليات أكثر من الموجودة بالفعل. فإذا أردت التأكد من استرجاع العمليات الحقيقية فقط، يمكنك اختبار الخاصية HasExited المصاحبة للعملية.

الوصول للملفات باستخدام الكائن My

يمكنك من خلال الكائن My والتى يمكنك من خلالها الوصول إلى الكائنات والخصائص والوظائف المختلفة شائعة الاستخدام داخل نطاق .Net المعروف باسم .Net Framework إلى جانب العناصر المختلفة التى قمت بإنشائها داخل التطبيق. كما يمكنك من خلال My أيضاً الوصول إلى الموارد المختلفة الموجودة على حاسبك كالحافظة Clipboard (من خلال الكائن My.clipboard) والمؤقت ولوحة المفاتيح والفأرة وغيرها. كما يمكنك كذلك من خلال My التعامل مع الملفات والعناصر المختلفة للشبكة. يوضح جدول ١٥-٣ التالى التصنيفات الأساسية داخل الكائن My واستخدام كلٍ منها.

جدول ١٥-٣ التصنيفات الأساسية داخل الكائن My

اسم التصنيف	الوظيفة
My.Application	يحتوى على بيانات عن التطبيق الحالى وخدماته المختلفة
My.Computer	يحتوى على ارتباطات مصادر وأدوات جهاز الكمبيوتر
My.Forms	يحتوى على ارتباطات لجميع النماذج الموجودة بالتطبيق

My.Log	الوصول إلى أدوات تسجيل سجل تطبيقات ASP.NET التى تستطيع التعامل مع عمليات الدخول بوسائل مختلفة
My.Request	الوصول إلى طلب الويب الحالى لتطبيقات ASP.NET
My.Resources	يحتوى على مصادر التطبيقات العامة
My.Response	الوصول إلى استجابة الويب الحالية مع تطبيقات ASP.NET
My.Settings	القدرة على تخزين إعدادات التطبيق الحالى واسترجاعها
My.User	الحصول على البيانات المختلفة لاسم المستخدم الحالى للتطبيق
My.WebServices	الوصول إلى خدمات الويب التى قام التطبيق بإنشاء مرجع لها

ويمكنك الوصول إلى الملفات المختلفة الموجودة على حاسبك من خلال الكائن My.Computer.FileSystem. فعلى سبيل المثال، لفتح أحد الملفات النصية من خلال المربع الحوارى Open وعرض محتوياته داخل مربع نص وليكن باسم txtFile، يمكنك استخدام الكود التالى:

```
Dim myText As String = " "
OpenFileDialog1.ShowDialog()
    If OpenFileDialog1.FileName <> " " Then
    myText = My.Computer.FileSystem.
ReadAllText(OpenFileDialog1.FileName)
      txtFile.Text = myText
    End If
```