

## الفصل الخامس عشر العمل مع الطباعة

تعرفنا فيما سبق على كيفية التعامل مع قواعد البيانات وإظهار البيانات المختلفة على الشاشة. سنقوم في هذا الفصل بالتعرف على كيفية إظهار هذه البيانات وغيرها على الورق من خلال التعرف على طرق وخصائص مختلفة خاصة بعملية الطباعة.

بانتهاء هذا الفصل ستتعرف على:

- تخطيط التقرير.
- إنشاء كائن عملية الطباعة.
- مفهوم أحداث الطباعة.
- ضبط الهوامش.
- تعيين الوسائل المساعدة للطباعة.

على الرغم من التقدم الكبير في كيفية عرض المعلومات سواءً من خلال الأجهزة الإلكترونية المختلفة أو من خلال شبكة الإنترنت، إلا أنك في جميع الأحوال في حاجة إلى البيانات المطبوعة التي يكون لها الرونق الخاص والمذاق المتميز. سنقوم في هذا الفصل والفصل التالي بإلقاء الضوء على كيفية إنشاء التقارير الورقية باستخدام كود **Visual Basic 2008** من خلال تقنيتين مختلفتين، الأولى من خلال المسمى **Printing** المصاحب لنماذج **Windows** والذي يتيح لك التحكم بشكل دقيق في العمليات المختلفة للطباعة وهو موضوع هذا الفصل، والثانية من خلال التقارير البلورية **Crystal Reports** وهي عبارة عن أداة متميزة لإنشاء التقارير تتيح لك إنشاء تقارير البيانات المركبة وهو موضوع الفصل القادم إن شاء الله.

يحتوي نطاق **.NET** على العديد من التصنيفات التي يمكنك استخدامها للتفاعل مع الطباعة المتصلة بجاسبك، حيث يمكنك طباعة النصوص والرسوم والتحكم في إعدادات الطباعة المختلفة مثل الهوامش ودرجة الوضوح. كما أن هناك العديد من التصنيفات التي تحتوي على مربعات حوارية تمكن المستخدم من اختيار الطباعة المناسبة أو عرض التقرير على الشاشة قبل إرساله إلى الطباعة. وتبنى عملية الطباعة داخل نماذج **Windows** أساساً على الرسوم، حتى إذا أردت طباعة النصوص فقط، فمازلت تخطط صفحات الطباعة باستخدام وظائف الرسوم وهي نفس الوظائف التي تستخدمها للرسم على النماذج والتي تقع داخل المسمى **System.Drawing.Graphics**. وتقع التصنيفات التي تحتوي على المهام الإضافية اللازمة لإرسال الرسوم للطباعة داخل المسمى **System.Drawing.Printing**. سنقوم في الفقرات التالية بالتعرف على بعض تصنيفات **.NET** المستخدمة لتسهيل عملية الطباعة من خلال تطبيق عملي على بعض الكتب الموجودة بشركة كمبيوساينس.

## تخطيط التقرير

بالمقارنة بالعمليات المختلفة التي تتم داخل حاسبك، يعتبر التقرير **Report** من الأشياء التي تحتاج إلى عناية خاصة كي تظهر بالمظهر المناسب والجذاب، لأنه مهما كانت قوة تطبيقك، فإن أحد الأهداف الرئيسية للتطبيق تتمثل في إظهار التقارير التي تفيدها فيما بعد في اتخاذ العديد من القرارات والخيارات الخاصة بتطوير المؤسسة ككل. لذا فغالباً لا تعنى مدير المؤسسة أو صاحبها قوة التطبيق بقدر ما يعنيه دقة التقارير وجمال تسيقها. فإذا أردت تمثيل التقرير باستخدام نماذج **Windows**، يجب أن تقوم بترجمة متطلبات التقرير إلى مجموعة من عمليات استدعاء وظائف الطباعة التي تقوم بإخبار الطباعة بتنفيذ هذه المتطلبات. لذا فمن الأفضل دائماً تقسيم متطلبات التقرير إلى قسمين أساسيين هما:

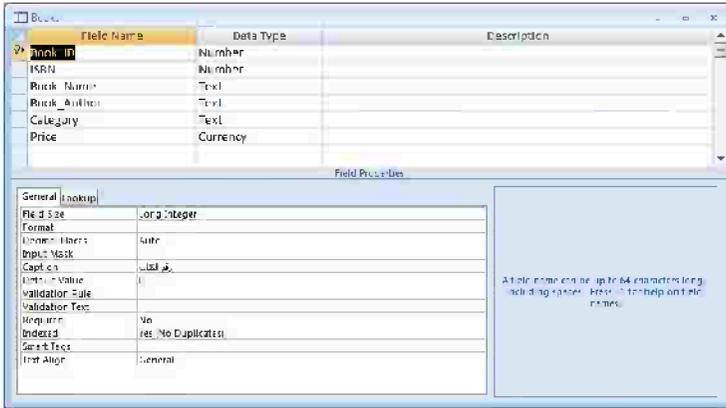
- بيانات التقرير **Report Data** حيث تبنى معظم التقارير على بيانات قاعدة بيانات. وقبل أن تقوم بكتابة سطر من الكود، يجب أن تعرف الحقول التي ترغب في تضمينها داخل التقرير بالإضافة إلى إنشاء عبارة **SQL** اللازمة لذلك.
- تخطيط التقرير **Report Layout** حيث تتشابه عملية تصميم التقرير كثيراً مع عملية تصميم النموذج، إلا أنك حينما تقوم بإنشاء التقرير بالكامل من خلال الكود، فلن تتمكن من مشاهدة النتائج لحظياً. وأثناء تخطيط التقرير، يمكنك تحديد عدد الأعمدة المستخدمة وكذلك تحديد رأس التقرير في كل صفحة من صفحاته والتي تحتوى غالباً على عنوان للتقرير بالإضافة إلى شعار الشركة أو المؤسسة. سنقوم فيما يلي بإنشاء تطبيق بسيط يتصل بقاعدة بيانات تحتوى على بيانات الكتب التي تتشرف شركة كمبيوساينس بنشرها ومن ثم يتم طباعة تقرير عن هذه الكتب ومن خلال هذا التطبيق سنقوم بشرح المهام والوظائف المختلفة للطباعة. تابع معنا الخطوات الآتية:

١. من داخل بيئة تطوير **Visual Basic 2008**، قم بإنشاء تطبيق نوافذ جديد باسم **Reports**.

٢. قم بإضافة زر أمر من مربع الأدوات إلى النموذج وقم بتغيير اسمه إلى `btnPrint` وعنوانه إلى "طباعة التقرير".
٣. افتح نافذة كود النموذج ثم قم بإدخال عبارتي التضمين التاليتين في أعلى تصنيف النموذج:

`Imports System.Drawing.Printing`  
`Imports System.Data.OleDb`

٤. قم بنسخ قاعدة البيانات `Compuscience.mdb` من القرص المدمج المرفق بالكتاب إلى مجلد التطبيق الحالي (`Reports`) حيث تحتوى قاعدة البيانات هذه على جدول باسم `Books` يحتوى على بيانات الكتب (انظر شكل ١٥-١).



شكل ١٥-١ محتويات الجدول `Books` داخل قاعدة البيانات

## إنشاء كائن عملية الطباعة

إذا كنت من مستخدمي الإصدارات السابقة من `Visual Basic`، فلاحظ أن لديك دراية بالتصنيف `Printer`. فعلى الرغم من التشابه الكبير بين استخدام هذا التصنيف ونماذج `Windows` في عملية الطباعة إلا أن هناك العديد من التغييرات الناتجة عن تأثير عملية الطباعة بالأحداث المختلفة حيث أصبحت مثل النماذج `Event Driven`. فبدلاً من إرسال جميع وثائقك من أولها إلى آخرها إلى الطابعة، يتم إرفاقها بكود به العديد من إجراءات الأحداث المختلفة حيث يقوم نطاق `.NET` باستدعاء هذه الإجراءات وقت

الحاجة. وعلى الرغم من اختلاف هذه الطريقة، إلا أنك تستطيع من خلالها التحكم في تنسيق التقرير بشكل أفضل من ذي قبل كما أنها تحتوي على بعض السمات الهامة الجديدة مثل معاينة صفحات الطباعة قبل إرسالها إلى الطباعة. أثناء العمل مع التطبيق الحالي، سنحتاج إلى إنشاء الإجراءات المخصصة التالية داخل تصنيف النموذج:

- الإجراء **PrintBooksReport** والذي يقوم باستهلال عملية الطباعة.
- الإجراء **PrintBooksPage** وهو عبارة عن إجراء حدثي يقوم بمعالجة حدث طباعة صفحة واحدة من التقرير.
- الإجراء **PrepareReportData** ويقوم بإجراء عملية الاتصال بقاعدة البيانات وإرجاع البيانات المطلوب عرضها داخل التقرير.
- الإجراء **ReportProcessed** وهو عبارة عن إجراء حدثي يتم استدعاؤه بمجرد انتهاء عملية الطباعة.

يمثل التصنيف **PrintDocument** عملية الطباعة داخل التطبيق. فإذا أردت إنشاء عملية طباعة جديدة، قم بإنشاء حالة جديدة من هذا التصنيف ثم قم بإضافة الإجراءات الحديثة المناسبة. للبدء في إنشاء التطبيق الحالي، قم بإضافة الإجراء التالي إلى تصنيف النموذج:

```
Private Sub PrintBooksReport ()  
    Dim BooksReport As PrintDocument  
    Dim dlgPreview As New PrintPreviewDialog()  
    BooksReport = New PrintDocument()  
  
    AddHandler BooksReport.PrintPage, AddressOf  
        PrintBooksPage  
    AddHandler BooksReport.BeginPrint, AddressOf  
        PrepareReportData  
    AddHandler BooksReport.EndPrint, AddressOf  
        ReportProcessed  
  
    BooksReport.Print()  
End Sub
```

يوضح هذا الإجراء حقيقةً الطريقة المتبعة دائماً في استخدام التصنيف **PrintDocument**. ففي البداية يتم استهلال الكائن **BooksReport** وبعد ذلك يتم

ربط الحدث **PrintPage** بمعالج الحدث المناسب (**PrintBooksPage**) وكذلك الحال بالنسبة لحدث بداية الطباعة ونهايتها وأخيراً يتم استدعاء الوظيفة (**Print**) لبدء عملية الطباعة.

يمكنك تعيين اسم مناسب للخاصية **DocumentName** ليظهر هذا الاسم داخل نافذة مدير الطباعة **Print Manager** والمربعات الحوارية الأخرى.



وبعد تعيين الدالة **PrintBooksReport**، قم باستدعاء هذه الدالة داخل حدث نقر الزر **btnPrint** كما يلي:

### Call **PrintBooksReport()**

وبهذا نكون قد أنشأنا الكود الخاص ببدء عملية الطباعة، والخطوة التالية هي إنشاء الكود اللازم لمعالجة أحداث هذه العملية.

## مفهوم أحداث الطباعة

حينما تقوم بإنشاء عملية الطباعة داخل نماذج **Windows**، يجب أن تقوم بكتابة الكود اللازم لطباعة صفحة واحدة في الوقت الحالي كاستجابة للحدث **PrintPage**. لاحتواء هذا الحدث داخل التطبيق الحالي، قم بإدخال الكود التالي الذي يقوم بمعالجة صفحة واحدة من تقرير الكتب المزمع إنشاؤه.

### Sub **PrintBooksPage(ByVal sender As Object, ByVal e As PrintPageEventArgs)**

```
Dim IDX As Integer = 100
Dim ISBNX As Integer = 200
Dim NameX As Integer = 300
Dim AuthorX As Integer = 500
Dim CategoryX As Integer = 600
Dim PriceX As Integer = 700
```

```
Dim FieldInfo As String
Dim RecordsPerPage As Integer = 20
Dim CurrentRecord As Integer = 0
Dim CurrentY As Integer = 300
Dim ReportFont As New Font("Arial", 12, FontStyle.Regular)
```

```
Dim ReportFontHeight As Integer =  
    ReportFont.GetHeight(e.Graphics)
```

```
While CurrentRecord < RecordsPerPage
```

```
    'Place information from current record on the page
```

```
    FieldInfo = rdrBooksInfo.GetInt32(0).ToString  
    e.Graphics.DrawString(FieldInfo, ReportFont,  
    Brushes.Black, IDX, CurrentY)
```

```
    FieldInfo = rdrBooksInfo.GetInt32(1).ToString.Trim  
    e.Graphics.DrawString(FieldInfo, ReportFont,  
    Brushes.Black, ISBNX, CurrentY)
```

```
    FieldInfo = rdrBooksInfo.GetString(2).Trim  
    e.Graphics.DrawString(FieldInfo, ReportFont,  
    Brushes.Black, NameX, CurrentY)
```

```
    FieldInfo = rdrBooksInfo.GetString(3).Trim  
    e.Graphics.DrawString(FieldInfo, ReportFont,  
    Brushes.Black, AuthorX, CurrentY)
```

```
    FieldInfo = rdrBooksInfo.GetString(4).Trim  
    e.Graphics.DrawString(FieldInfo, ReportFont,  
    Brushes.Black, CategoryX, CurrentY)
```

```
    FieldInfo = rdrBooksInfo.GetDecimal(5).ToString  
    e.Graphics.DrawString(FieldInfo, ReportFont,  
    Brushes.Black, PriceX, CurrentY)
```

```
    CurrentY = CurrentY + ReportFontHeight
```

```
    'Move to the next record
```

```
    If Not rdrBooksInfo.Read() Then
```

```
        Exit While
```

```
    End If
```

```
    CurrentRecord += 1
```

```
End While
```

```
If CurrentRecord < RecordsPerPage Then
```

```
    e.HasMorePages = False
```

```
Else
```

```
    e.HasMorePages = True
```

End If

End Sub

وعن هذا الكود، نوضح ما يلي:

- يستخدم الكود السابق في احتواء (معالجة) الحدث **PrintPage** الذى يقوم التصنيف **PrintDocument** بالاحتفاظ به لطباعة جميع صفحات التقرير حتى يتم تخصيص القيمة **False** للخاصية **HasMorePages** المصاحبة للمعامل الثانى للإجراء (المعامل **e**) والذى يحتوى بدوره على التصنيف **Graphics** الذى يستخدمه الكود في تفسير الصفحة المطبوعة.
- كلما قمت برسم سطر من النصوص باستخدام الكائن **e.Graphics**، يجب أن تقوم بتحديد مكان هذا السطر باستخدام الإحداثيات **X,Y**. لذا فعندما تقوم بوضع الحقول النصية على الصفحة، يتم استخدام متغير عداد باسم **CurrentY** لتعيين المكان الرأسى لبداية كتابة النص. كما يتم استخدام الوظيفة **Font.GetHeight()** لإرجاع ارتفاع الخط تبعاً للخط المستخدم.
- عند كتابة كود معالجة الحدث **PagePrint** يكون لديك الخيار في تحديد البيانات التى تحتاج إلى طباعتها داخل التقرير.
- يحتوى الكود على كائن التصنيف **OleDbDataReader** الذى يحتوى على وظيفة للتقدم إلى السجل التالى داخل قاعدة البيانات. ولكن إذا كنت تستخدم مصفوفة بدلاً من قاعدة البيانات، يمكنك في هذه الحالة استخدام متغير يحتوى على الدليل الحالى داخل المصفوفة.
- إذا كنت ترغب في طباعة تقرير يتكون من صفحة واحدة فقط، يمكنك تخصيص القيمة **False** للخاصية **HasMorePages** في نهاية إجراء معالجة الحدث.

استخدام أحداث الطباعة الأخرى

تحتاج على الأقل إلى كتابة كود الحدث **PrintPage** لتحديد محتويات كل صفحة من صفحات التقرير المثلة في الكائن **PrintDocument**. لكن يوجد حدثان إضافيان

يمكنك من خلالهما تعيين مهام الاستهلال والتنظيف التي تتم قبل بداية عملية الطباعة وبعد نهايتها على الترتيب وهما الحدث **BeginPrint** والحدث **EndPrint**. ولأننا اعتبرنا في الكود السابق أن هناك كائن **OleDbDataReader** مفتوح، فخير مكان لهذا الكائن هو إجراء معالجة الحدث **BeginPrint** والذي يتم من خلاله الاتصال بقاعدة البيانات **Compuscience** وفتح هذا الكائن تمهيداً لقراءة بيانات التقرير. كما يمكنك استخدام الحدث **EndPrint** لقطع عملية الاتصال بعد الانتهاء من طباعة بيانات التقرير. قم الآن بكتابة كود هذه الأحداث كما يلي:

```
Dim rdrBooksInfo As OleDbDataReader
Dim con As OleDbConnection
Private Sub PrepareReportData(ByVal sender As Object, ByVal e
As PrintEventArgs)
    Dim sSQL As String
    Dim sConnectionString As String
    Dim cmd As OleDbCommand

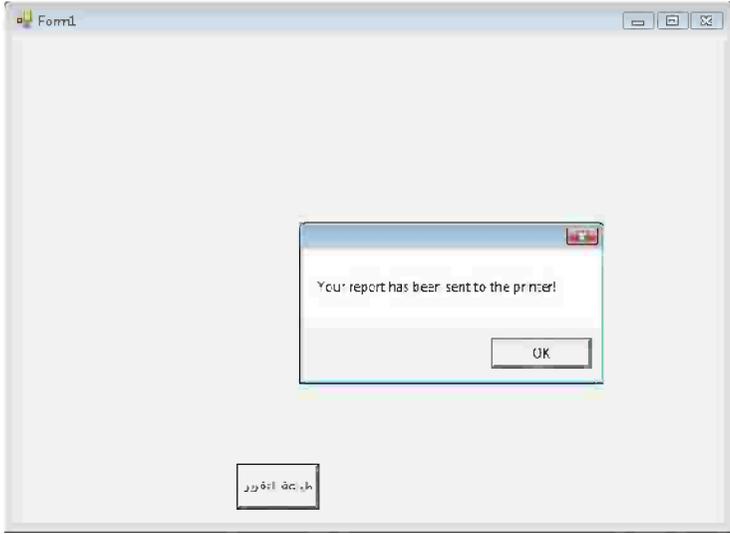
    'Open OleDbDataReader and position to first record
    sConnectionString = "Provider= Microsoft.Jet.OLEDB.4.0; Data
        Source = ..\Compuscience.mdb"
    con = New OleDbConnection(sConnectionString)
    sSQL = "SELECT * FROM Books Order by Book_ID"
    cmd = New OleDbCommand(sSQL, con)
    con.Open()
    rdrBooksInfo = cmd.ExecuteReader()
    rdrBooksInfo.Read()
End Sub
Private Sub ReportProcessed(ByVal sender As Object, ByVal e
As PrintEventArgs)
    con.Close()
    MessageBox.Show("Your report has been sent to the
        printer!")
End Sub
```

وعن هذا الكود، نوضح ما يلي:

- قمنا بالإعلان عن الكائن `rdrBooksInfo` داخل مستوى التصنيف نفسه حتى يتم الوصول إليه من إجراء احتواء الحدث `PrintPage`.
- تقوم الدالة الأولى `PrepareReportData` بإنشاء الكائن `OleDbDataReader` ووضعه على السجل الأول داخل الجدول.
- تقوم الدالة الثانية بإغلاق عملية الاتصال بقاعدة البيانات مع إظهار رسالة للمستخدم توضح انتهاء عملية طباعة التقرير.
- حتى يتم استدعاء الدالتين في التوقيت المناسب، يجب ربطهما بالأحداث المصاحبة داخل التصنيف `PrintDocument` وهو ما فعلناه من قبل بالدالة `PrintBooksReport`.
- يحتوى المعامل الثانى لإجراء معالجة الحدث `BeginPrint` والحدث `EndPrint` على كائن من النوع `PrintEventArgs` والذي يختلف عن معامل إجراء الحدث `PrintPage` (الكائن `PrintPageEventArgs`)، حيث يحتوى هذا الكائن على الخاصية `CancelPrint` التى يمكنك استخدامها لإلغاء عملية الطباعة لسبب من الأسباب.

## اختبار التطبيق

إذا قمت بإدخال الكود السابق، تأكد من اتصال طابعتك بالحاسب ثم قم بتشغيل التطبيق وانقر زر "طباعة التقرير" للبدء فى طباعة تقرير عن بيانات الكتب الموجودة بقاعدة البيانات. فإذا تم كل شئ على ما يرام، ستظهر لك رسالة تحبرك بإرسال التقرير إلى الطابعة (انظر شكل ١٥-٢).



شكل ١٥-٢ إرسال التقرير إلى الطباعة

## ضبط المواضع

إذا تابعت معنا المثال السابق وقمت بطباعة التقرير الذى يحتوى على قائمة الكتب التى تنشرها شركة كمبيوساينس، لوجدت مدى سهولة تخطيط التقرير. فكل ما هو مطلوب منك هو تعيين الإحداثيين الأفقى والرأسى لكل عنصر من عناصر التقرير. وبالرجوع إلى الكود السابق، تجد أننا استخدمنا الوظيفة `Graphics.DrawString` لكتابة نص الكائن الرسومى عند الإحداثى الأفقى `X` والرأسى `Y`. وعلى الرغم من الحصول على قيمة الإحداثى `Y` تلقائياً من خلال العداد، إلا أننا قمنا بكتابة كود الإحداثى الرأسى `X` لكل عمود من النصوص باستخدام قيم ثابتة كما يلي:

```
Dim IDX As Integer = 100
Dim ISDNX As Integer = 200
Dim NameX As Integer = 300
Dim AuthorX As Integer = 500
Dim CategoryX As Integer = 700
Dim PriceX As Integer = 900
```

ويتم قياس هذه القيم بوحدة تسمى **Display** والتي تساوى ٧٥/١ من البوصة. وربما تعمل هذه الطريقة مع بعض التقارير والطابعات، إلا أنها كما ترى تفتقد إلى المرونة بشكل كبير.

لتتعرف على المزيد عن الأنواع المختلفة لوحدات القياس المتاحة للاستخدام مع وظائف الرسوم، قم بالبحث عن العنوان **GraphicsUnit** داخل ملفات المساعدة المصاحبة لبرنامج **Visual Studio 2008**.



يحتوى المعامل الثانى داخل إجراء الحدث **PrintPage** على العديد من الخصائص التى يمكنك استخدامها لوضع الكائنات مقارنةً بحدود وهوامش الصفحة، ومن أهمها الخاصيتين التاليتين:

- الخاصية **MarginBounds** وتستخدم لتحديد هوامش الصفحة الحالية.
- الخاصية **PageBounds** وتستخدم لتحديد حدود الصفحة الحالية وبالتالي توسيط الكائنات داخل الصفحة وإنشاء حواشيتها السفلية.

وتقوم كل من الخاصيتين بإرجاع الكائن **Rectangle** الذى يحتوى بدوره على الخصائص المستخدمة لتحديد مكان الكائن وحجمه. يوضح الكود التالى كيفية استخدام هذه الخصائص عوضاً عن القيم الثابتة التى قمنا بتحديددها فى الكود السابق:

```
Private Function GetColX(ByVal ColNum As Integer, ByVal Boundary As Rectangle) As Single
```

```
Dim colWidthPct() As Integer = {10,10,30,20,20,10}
```

```
If ColNum = 0 Then
```

```
'The first column is placed at the page margin
```

```
Return Boundary.X
```

```
Else
```

```
'X is at the end of the previous column
```

```
Dim CurrentX As Single
```

```
Dim i As Integer
```

```
For i = 0 To ColNum - 1
```

```
CurrentX = CurrentX + (colWidthPct(i) / 100) *
```

## Boundary.Width

Next  
Return CurrentX  
End If  
End Function

في هذا الكود قمنا بتعريف عرض الأعمدة الستة كنسبة من عرض الصفحة داخل المصفوفة `colWidthPct`. وعند نداء هذه الدالة، يتم تحرير رقم العمود داخل الأعمدة الستة (أى من 0 إلى 5) بالمعامل الأول، وقيمة الخاصية `MarginBounds` بالمعامل الثاني كما يلي:

**NameX = GetColX(0, e.MarginBounds)**

وعلى الرغم من أن هذه الدالة تساعدك على تحديد الإحداثى الأفقى `X`، إلا أنها لا تقوم بإرجاع عرض العمود كما أنها لا تقوم أيضاً بمنع النص من تعدى حدود هذا العمود. إلا أن هذه الطريقة أفضل في جميع الأحوال من الطريقة السابقة التى نقوم فيها بتحديد الإحداثى الأفقى صراحةً.

## تعيين الوسائل المساعدة للطباعة

على الرغم من تناولنا حتى الآن للتقنيات اللازمة لإرسال البيانات إلى الطباعة، فإن لدى معظم المستخدمين الطموح في مزيدٍ من التحكم في عملية الطباعة أو على الأقل القدرة على إلغاء طلب الطباعة في أى وقتٍ من الأوقات طالما أن التقرير مازال في انتظار دوره في طابور الطباعة. كما يجب أيضاً إضافة العديد من السمات المشهورة والمعروفة لمستخدمى **Windows** والتي من أهمها ما يلي:

- معاينة الطباعة **Print Preview** والتي تتيح للمستخدم معاينة الوثيقة على الشاشة قبل إرسالها إلى الطباعة.
- اختيار الطباعة **Printer Selection** والتي تتيح للمستخدم اختيار الطباعة المناسبة من الطابعات المختلفة المتصلة بحاسبه.
- إعداد الصفحة **Page Setup** والتي تتيح للمستخدم تغيير هوامش الصفحة وغيرها من الإعدادات الأخرى التى من شأنها التحكم في جودة الطباعة.

وباستخدام السمات السابقة، يمكنك تحديد مقدار التحكم الذى ترغب فى إعطائه للمستخدم سواء بتوفير المربعات الحوارية أو تعيين الخيارات والخصائص بنفسك من خلال الكود. فهناك بعض البرامج مثل برنامج **Microsoft Word** والذى تمكن المستخدم من الطباعة المباشرة باستخدام الخيارات الافتراضية من خلال زر الطباعة الموجود بشريط الأدوات القياسى أو التحكم فى الطباعة وتغيير خصائص الصفحات المطبوعة من خلال الخيار **Print** داخل قائمة **File**.

### إضافة معاينة الطباعة

يحتوى نطاق **NET**. على تصنيفين لتمكينك من إضافة سمة معاينة الطباعة إلى تطبيقك وهما:

التصنيف **PrintPreviewDialog** ويمكنك من خلاله معاينة التقرير داخل نافذة مستقلة تحتوى على شريط للأدوات والذى يحتوى بدوره على مجموعة من الأزرار المستخدمة لتغيير حجم العرض واختيار الصفحات والطباعة. التصنيف **PrintPreviewControl** ويمكنك من خلاله معاينة التقرير داخل نفس النموذج.

ويتم تمثيل كل من التصنيفين بأداة تحكم داخل مربع الأدوات، إلا أنه يمكنك بسهولة استخدام التصنيف **PrintPreviewDialog** من داخل الكود. لتوضيح ذلك، سنقوم بإضافة سمة معاينة التقرير قبل طباعته إلى التطبيق الذى بين يدينا وذلك باستخدام التصنيف **PrintPreviewDialog**. تابع معنا الخطوات الآتية:

١. قم بإضافة زر أمر من مربع الأدوات إلى نموذج التطبيق. أعد تسمية الزر إلى **btnPreview** وعنوانه إلى "معاينة الطباعة".

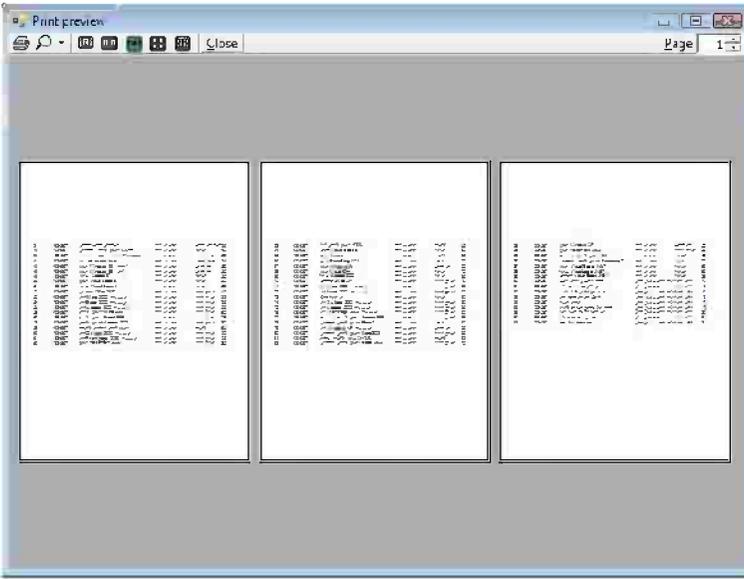
٢. انقر الزر الجديد نقرأ مزدوجاً ثم قم بإضافة الكود التالى إلى إجراء حدث نقر الزر:

```
Private Sub btnPreview_Click(ByVal sender As System.Object,
ByVal e As System.EventArgs) Handles btnPreview.Click
    Dim BooksReport As PrintDocument
    Dim dlgPreview As New PrintPreviewDialog()
```

```
BooksReport = New PrintDocument()  
dlgPreview = New PrintPreviewDialog()  
AddHandler BooksReport.PrintPage, AddressOf  
PrintBooksPage  
AddHandler BooksReport.BeginPrint, AddressOf  
PrepareReportData
```

```
dlgPreview.Document = BooksReport  
dlgPreview.ShowDialog()  
End Sub
```

٣. قم بتشغيل البرنامج ثم انقر زر "معاينة الطباعة"، تظهر نافذة المعاينة والتي تشبه إلى حد كبير تلك الموجودة بشكل ١٥-٣.



شكل ١٥-٣ معاينة التقرير قبل طباعته من خلال نافذة Print Preview

لاحظ أننا في الكود السابق لم نقم بتعيين الوظيفة **Print()** لطباعة التقرير بعد معاينته، وإنما يقوم الكائن **dlgPreview** بأداء هذه العملية تلقائياً حينما يقوم المستخدم بنقر زر الطباعة من شريط أدوات نافذة المعاينة، الذي يحتوي بدوره على العديد من السمات المفيدة مثل تكبير المعاينة أو تصغيرها وكذلك الانتقال بين صفحات التقرير.

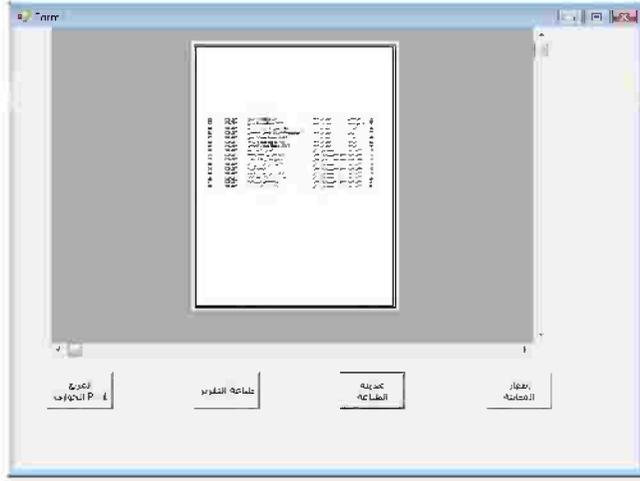
رأينا في هذه الطريقة عرض نافذة المعاينة بمعزلٍ عن التطبيق. لكن أحياناً ترغب في تضمين نافذة المعاينة داخل نفس النموذج بدلاً من فتح نموذج آخر أو نافذة أخرى. يمكنك في هذه الحالة استخدام التصنيف **PrintPreviewControl** بدلاً من التصنيف **PrintPreviewDialog**. للتعرف على طريقة استخدام هذا التصنيف من خلال أدوات المصاحبة الموجودة بمربع الأدوات، تابع معنا الخطوات الآتية:

١. قم بإضافة أداة **PrintPreviewControl** من مربع الأدوات إلى النموذج.
٢. قم بضبط حجم الأداة كي تتناسب مع البيانات التي سيتم عرضها.
٣. قم بإضافة زر جديد من مربع الأدوات إلى النموذج مع تغيير اسمه إلى **btnPrevControl** وعنوانه إلى "إظهار المعاينة".
٤. انقر الزر الجديد نقرأ مزدوجاً ثم قم بإدخال الكود التالي وذلك بتخصيص كائن **PrintDocumen** لأداة **PrintPreviewControl** ثم استدعاء الوظيفة **Show()** لإظهار معاينة الطباعة على نفس النموذج:

```
Private Sub btnPrevControl_Click(ByVal sender As System.Object, ByVal e As System.EventArgs) Handles btnPrevControl.Click
    Dim BooksReport As PrintDocument
    BooksReport = New PrintDocument()

    AddHandler BooksReport.PrintPage, AddressOf PrintBooksPage
    AddHandler BooksReport.BeginPrint, AddressOf PrepareReportData
    PrintPreviewControl1.Document = BooksReport
    PrintPreviewControl1.Show()
End Sub
```

٥. قم بتشغيل التطبيق ثم انقر زر "إظهار المعاينة"، تلاحظ ظهور معاينة الطباعة داخل الأداة **PrintPreviewControl** بالنموذج (انظر شكل ١٥-٤).



شكل ١٥-٤ استخدام أداة معاينة الطباعة لإظهار المعاينة داخل نفس النموذج

تقوم الأداة **PrintPreviewControl** بعرض بيانات المعاينة فقط، فهي لا تحتوي على شريط للأدوات، لذا يجب أن تقوم بتحديد خصائص صفحة المعاينة مثل الحجم والعمليات الأخرى من خلال الكود. وبالإضافة إلى ذلك، يمكنك توفيق الأداة لتحسين مظهرها باستخدام عدد من الخصائص وعلى رأسها ما يلي:

- خصائص الأعمدة والصفوف **Columns** و **Rows** وتحدد عدد الصفحات التي تظهر داخل الأداة في نفس الوقت.
- الخاصية **Startpage** وتحدد الصفحة الافتراضية التي تظهر داخل الأداة عند تشغيل التطبيق أو الصفحة التي تظهر بالركن الأيسر العلوي إذا احتوت الأداة على أكثر من صفحة.
- الخصائص **Zoom** و **AutoZoom** وتحدد حجم التقرير داخل نافذة المعاينة.

وعلى الرغم من عدم احتواء التصنيف **PrintPreviewControl** على الوظيفة **Print()** المستخدمة لبدء عملية الطباعة، إلا أنك تستطيع استخدام نفس الوظيفة المصاحبة للكائن **PrintDocument** المصاحب بدوره للأداة من خلال الخاصية **Document** لإرسال التقرير إلى الطابعة كما في الكود التالي:

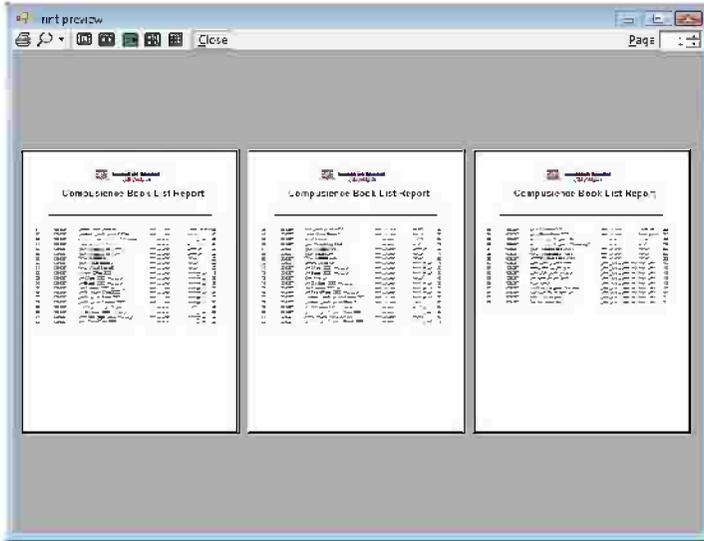
```
PrintPreviewControl1.Document.Print()
```

### إضافة رأس التقرير

يمكنك إضافة ما تريد إلى رأس التقرير بسهولة شديدة من خلال وظائف الرسم المختلفة. لتوضيح ذلك، سنقوم بإضافة شعار شركة كمبيوساينس وعنوان التقرير وكذلك خط بعرض الصفحة إلى رأس التقرير. لأداء ذلك قم بإضافة الإجراء التالي إلى تصنيف النموذج ثم قم باستدعائه قبل إظهار بيانات الحقول داخل الإجراء PrintBooksPage:

```
Private Sub SetupHeader(ByRef gx As Graphics)
    Dim sPictureFile As String
    sPictureFile = Application.StartupPath & "\..logo.jpg"
    If System.IO.File.Exists(sPictureFile) Then
        gx.DrawImage(New Bitmap(sPictureFile), 150, 125)
    End If
    gx.DrawString("Compusience Book List Report", New
        Font("Arial", 26, FontStyle.Bold), Brushes.Black, 150, 145)
    gx.DrawLine(New Pen(Brushes.Black), 100, 250, 750, 250)
End Sub
```

قم الآن بتشغيل التطبيق مرةً أخرى ثم انقر زر "معاينة الطباعة"، تلاحظ ظهور صورة الشعار وعنوان التقرير والخط الأفقي برأس التقرير (انظر شكل ١٥-٥).



شكل ١٥-٥ نافذة المعاينة بعد إضافة رأس التقرير

### التحكم في إعدادات الصفحة

كما ذكرنا من قبل، يمكنك كتابة الكود الذى يأخذ فى الاعتبار هوامش وحدود الصفحة. ولكن أحياناً ترغب فى توجيه التحكم فى هذه الخصائص للمستخدم نفسه. ويتيح لك نطاق .NET بعض التصنيفات التى تمكنك من التحكم فى خصائص الصفحة من خلال الكود أو من خلال المربعات الحوارية التى يكون القرار الأول فيها للمستخدم. يوضح الكود التالى كيفية استخدام الطريقتين، حيث يقوم أولاً بتعيين اتجاه الصفحة وهوامشها وبعد ذلك يتم إظهار المربع الحوارى Page Setup الذى يتيح للمستخدم تغيير هذه الإعدادات:

```
Dim MySettings As PageSettings
Dim UserSetup As PageSetupDialog
```

تعيين إعداداتك الافتراضية،

```
MySettings = New PageSettings()
MySettings.LandScape = True
MySettings.Margins = New Margins(150,150,150,150)
```

إظهار مربع حوارى لتمكين المستخدم من تغيير الإعدادات الافتراضية،

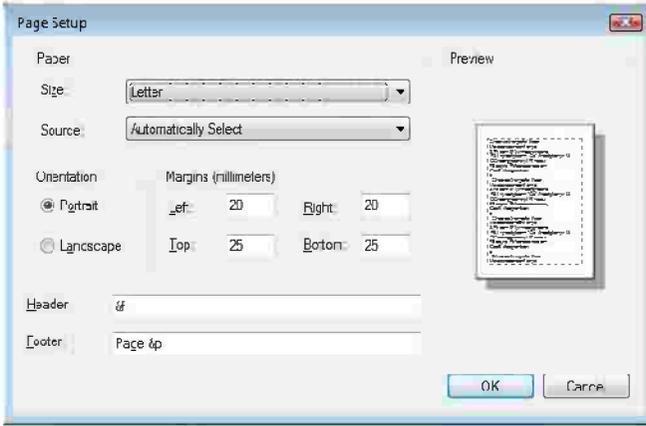
```
UserSetup = New PageSetupDialog()
UserSetup.PageSettings = MySettings
UserSetup.ShowDialog()
```

تخصيص إعدادات المستخدم إلى التقرير،

```
BooksReport.DefaultPageSettings = MySettings
```

وكما ترى فقد قمنا بتخصيص الإعدادات الحالية إلى المربع الحوارى من خلال الخاصية PageSettings ثم تخصيص خيارات المستخدم إلى التقرير مرةً أخرى من خلال الخاصية DefaultPageSettings. كما قمنا فى هذا الكود باستخدام التصنيف Margins المستخدم لتحديد هوامش الصفحة، حيث يتم تعيين هوامش بمقدار ١٠٠/١ من البوصة. ففى الكود السابق على سبيل المثال قمنا بتعيين هامش مقداره ١.٥ بوصة فى جميع الاتجاهات.

إذا قمت بتنفيذ الكود السابق من خلال زر أمر جديد بنموذج التطبيق الحالي، تحصل على المربع الحوارى **Page Setup** الذى تم تقسيمه إلى مجموعة من الأجزاء تشتمل على الهوامش واتجاه الصفحة وحجم الورق المستخدم فى عملية الطباعة ومصدره (انظر شكل ٦-١٥).



شكل ٦-١٥ المربع الحوارى **Page Setup**

يمكنك تقييد المستخدم من تغيير أى من هذه الإعدادات من خلال الكود. فمثلاً يستخدم الكود التالى لمنع المستخدم من تعديل اتجاه ورقة الطباعة داخل المربع الحوارى **Page Setup** مع الإبقاء على فرصة تغييره من داخل الكود:

**UserSetup.AllowOrientation = False**

#### تعيين إعدادات الطباعة

تختلف إعدادات الطباعة من طابعة إلى أخرى، فلكلٍ منها خصائصها المستقلة. يمكنك التحكم فى إعدادات الطباعة بالإضافة إلى إمكانية قائمة بالطابعات المثبتة على حاسبك من خلال الكود باستخدام التصنيف **PrinterSettings**. يوضح الكود التالى كيفية إظهار جميع الطابعات المثبتة على حاسبك مع إظهار رسالة بإمكانيات طباعة الألوان فى كلٍ منها:

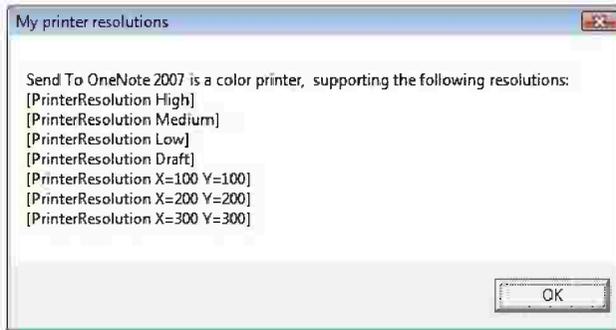
```
Dim MySettings As New PrinterSettings()
Dim PrinterNameList As PrinterSettings.StringCollection
Dim CurrentPrinterName As String
Dim i As Integer
```

## Dim s As String

```
PrinterNameList = PrinterSettings.InstalledPrinters
For Each CurrentPrinterName In PrinterNameList
    MySettings.PrinterName = CurrentPrinterName
    If MySettings.SupportsColor Then
        s = CurrentPrinterName & " is a color printer, "
        s &= " supporting the following resolutions:" & vbCrLf
        For i = 0 To MySettings.PrinterResolutions.Count - 1
            s &= MySettings.PrinterResolutions(i).ToString & vbCrLf
        Next
    Else
        s = CurrentPrinterName & " is not a color printer."
    End If

    MessageBox.Show(s,"My printer resolutions")
Next
End Sub
```

يحتوى التصنيف `PrinterSettings` على خاصية ساكنة واحدة هي الخاصية `InstalledPrinters` التى يمكنك استخدامها للحصول على قائمة بأسماء الطابعات المثبتة على الحاسب. فإذا أردت سمات وإعدادات كل طابعة على حده، يجب أن تقوم أولاً بتخصيص الطابعة لحالة من التصنيف `PrinterSettings` من خلال الخاصية `PrinterName`. وقد قمت بتجربة هذا الكود على حاسبى، فحصلت على الرسالة الموضحة فى شكل ٧-١٥ التالى.



شكل ٧-١٥ يحتوى الرسالة على الطابعة المثبتة على حاسبى وإعدادات ألوانها

### إظهار المربع الحوارى *Print*

بدلاً من تعيين الخصائص واحدةً تلو الأخرى من خلال الكائن **PrinterSettings**، يمكنك نقل الدفعة إلى المستخدم لتعيين هذه الخصائص من خلال المربع الحوارى **Print**، الذى يستطيع المستخدم من خلاله اختيار الطابعة المناسبة وعدد نسخ التى يرغب فى طباعتها وغيرها من الخيارات الأخرى. لإظهار المربع الحوارى **Print**، قم أولاً بإنشاء حالة جديدة من التصنيف **PrintDialog** ثم قم بربطه بالتقرير المطلوب من خلال الخاصية **Document** وأخيراً قم باستدعاء الوظيفة **ShowDialog** كما فى الكود التالى:

```
Dim BooksReport As PrintDocument
Dim PrinterSetupScreen As PrintDialog
Dim ButtonPressed As DialogResult
```

إعداد التقرير'

```
BooksReport = New PrintDocument()
```

```
AddHandler BooksReport.PrintPage, AddressOf PrintBooksPage
```

```
AddHandler BooksReport.BeginPrint, AddressOf
```

```
PrepareReportData
```

```
AddHandler BooksReport.EndPrint, AddressOf ReportProcessed
```

إظهار المربع الحوارى **Print**'

```
PrinterSetupScreen = New PrintDialog()
```

```
PrinterSetupScreen.Document = BooksReport
```

```
ButtonPressed = PrinterSetupScreen.ShowDialog()
```

قم بالطباعة إذا قام المستخدم بنقر زر **Ok** '

```
If ButtonPressed = DialogResult.OK Then
```

```
BooksReport.Print()
```

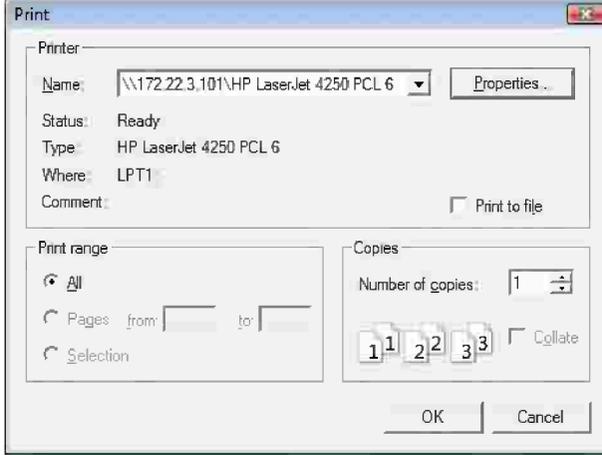
```
End If
```

وكما هو الحال مع التصنيف **PageSettingsDialog**، يحتوى التصنيف

**PrintDialog** على العديد من الخصائص التى يمكنك استخدامها لتوفيق المربع الحوارى

**Print** كيفما تشاء. يمكنك على سبيل المثال تمكين المستخدم من تحديد مدى من

الصفحات التي يتم طباعتها. إذا قمت بإضافة الكود السابق إلى إجراء حدث نقر أحد الأزرار الجديدة بالنموذج، ستحصل على نتيجة مشابهة لشكل ٨-١٥ التالي.



شكل ٨-١٥ المربع الحوارى Print

