**مواضيع متقدّمة**

ليس معنى أنّ هذه مواضيع متقدّمة أنّها أصعب من غيرها، بل معناه فقط أنّها مواضيعُ يمكن تأخير تعلّمها لفترة، لأنّك لن تحتاجَ إليها بصورةٍ ملحّة في تطبيقاتك.

**11.** [**أدوات متقدّمة Advanced Controls:**](file:///C:\Users\Administrator\Desktop\vb.net\d-%20%20مواضيع%20متقدّمة\11-%20أدوات%20متقدّمة\03-%20مواضيع%20متقدّمة\11-%20أدوات%20متقدّمة.htm)

التعرّف على خصائص ووسائل وأحداث مربّع النصّ المنسّق RichTextBox، وقائمة الصور ImageList، والشجرة TreeVeiw، وقائمة العرض ListView.

**12.** [**الرسم والتلوين:**](file:///C:\Users\Administrator\Desktop\vb.net\d-%20%20مواضيع%20متقدّمة\11-%20أدوات%20متقدّمة\03-%20مواضيع%20متقدّمة\12-%20الرسم%20والتلوين.htm)

التعرّف على كيفيّة التعامل مع الألوان والخطوط والصور ومربّع الصورة BictureBox، والإمكانيّات المتقدّمة التي يوفّرها لنا GDI+.

**13.** [**بناء الأدوات الخاصّة:**](file:///C:\Users\Administrator\Desktop\vb.net\d-%20%20مواضيع%20متقدّمة\11-%20أدوات%20متقدّمة\03-%20مواضيع%20متقدّمة\13-%20بناء%20الأدوات%20الخاصّة.htm)

- التعرّف على كيفيّة وراثة الأدوات الموجودة وتطويرها.

- التعرّف على كيفيّة إنشاء أدوات جديدة تتكوّن من مجموعة من الأدوات الموجودة.

- التعرّف على كيفيّة إنشاء أدوات ذات واجهة استخدام نرسمها نحن.

- التعرّف على كيفيّة التحكّم في ألوان وخطوط وأبعاد عناصر القوائم الرئيسيّة Menus وقوائم العناصر Lists.

**14.** [**المهامّ المتعدّدة Multithreading:**](file:///C:\Users\Administrator\Desktop\vb.net\d-%20%20مواضيع%20متقدّمة\11-%20أدوات%20متقدّمة\03-%20مواضيع%20متقدّمة\14-%20Multithreading%20المهامّ%20المتعدّدة.htm)

التعرّف على كيفيّة تقسيم البرنامج إلى عمليّات مستقلّة، وكيفيّة التحكّم في هذه العمليّات.

**الفصل الحادي عشر**

**أدوات ويندوز متقدّمة**

**·** [**مربّع النصّ المنسّق RichTextBox Control:**](file:///C:\Users\Administrator\Desktop\vb.net\d-%20%20مواضيع%20متقدّمة\11-%20أدوات%20متقدّمة\11-%20أدوات%20متقدّمة\1-%20مربّع%20النصّ%20المنسّق.htm)

**·** [**قائمة الصور ImageList Control:**](file:///C:\Users\Administrator\Desktop\vb.net\d-%20%20مواضيع%20متقدّمة\11-%20أدوات%20متقدّمة\11-%20أدوات%20متقدّمة\2-%20قائمة%20الصور.htm)

**·** [**أداة العرض المتفرّع "الشجرة" TreeView Control:**](file:///C:\Users\Administrator\Desktop\vb.net\d-%20%20مواضيع%20متقدّمة\11-%20أدوات%20متقدّمة\11-%20أدوات%20متقدّمة\3-%20الشجرة.htm)

**·** [**قائمة العرض:**](file:///C:\Users\Administrator\Desktop\vb.net\d-%20%20مواضيع%20متقدّمة\11-%20أدوات%20متقدّمة\11-%20أدوات%20متقدّمة\4-%20قائمة%20العرض.htm)

**·** [**السحبُ والإدراج:**](file:///C:\Users\Administrator\Desktop\vb.net\d-%20%20مواضيع%20متقدّمة\11-%20أدوات%20متقدّمة\11-%20أدوات%20متقدّمة\5-%20السحبُ%20والإدراج.htm)

**مربّع النصّ المنسّق RichTextBox Control**

* [**ما هو:**](file:///C:\Users\Administrator\Desktop\vb.net\d-%20%20مواضيع%20متقدّمة\11-%20أدوات%20متقدّمة\1-%20مربّع%20النصّ%20المنسّق\1-%20ما%20هو%20مربّع%20النصّ%20المنسّق.htm)
* [**لغة النصوص المنسّقة RTF Language:**](file:///C:\Users\Administrator\Desktop\vb.net\d-%20%20مواضيع%20متقدّمة\11-%20أدوات%20متقدّمة\1-%20مربّع%20النصّ%20المنسّق\2-%20لغة%20النصوص%20المنسّقة.htm)
* [**مشروع RTFDemo:**](file:///C:\Users\Administrator\Desktop\vb.net\d-%20%20مواضيع%20متقدّمة\11-%20أدوات%20متقدّمة\1-%20مربّع%20النصّ%20المنسّق\3-%20مشروع%20النصّ%20المنسّق.htm)
* [**خصائص مربّع النصّ المنسّق:**](file:///C:\Users\Administrator\Desktop\vb.net\d-%20%20مواضيع%20متقدّمة\11-%20أدوات%20متقدّمة\1-%20مربّع%20النصّ%20المنسّق\4-%20خصائص%20مربّع%20النصّ%20المنسّق.htm)
* [**وسائل مربّع النصّ المنسّق:**](file:///C:\Users\Administrator\Desktop\vb.net\d-%20%20مواضيع%20متقدّمة\11-%20أدوات%20متقدّمة\1-%20مربّع%20النصّ%20المنسّق\5-%20وسائل%20مربّع%20النصّ%20المنسّق.htm)
* [**أحداث مربّع النصّ المنسّق:**](file:///C:\Users\Administrator\Desktop\vb.net\d-%20%20مواضيع%20متقدّمة\11-%20أدوات%20متقدّمة\1-%20مربّع%20النصّ%20المنسّق\6-%20أحداث%20مربّع%20النصّ%20المنسّق.htm)

**ما هو مربّع النصّ المنسّق:**

إنّ الترجمة الحرفيّة تقتضي أن نسمّيَ هذه الأداة "مربّع النصّ الثريّ"، وهو فعلا ثريّ بالإمكانيّات التي تتيح لك تنسيق النصّ بطريقة تقارب ما تراه في وثائق Microsoft Word.

إنّ هذه الأداة تمتلك جميع قدرات مربّع النصّ العاديّ، بالإضافة لقدرتك على اختيار خطّ وحجم ولون وهامش أيّ جزء من النصّ على حدة، كما يمكنك إضافة الصور داخل النصّ.

ولكن كيف يمكن كتابة نصوص لها تنسيقات مختلفة داخل مربّع نصّ واحد؟

إنّ فعل ذلك يتطلّب لغة خاصّة نكتب بها النصّ ومعلومات تنسيقه.. هذه اللغة تسمّى RTF أي "تنسيق النصّ الثريّ" Rich Text Format.. وقبل أن ينقبضَ صدرك وتتخيّل أنّك بصدد تعلّم لغة جديدة، يجب أن أخبرك أنّك لن تفعل هذا أبدا.. إنّ أداة مربّع النصّ المنسّق موجودة لتريحك من هذا العناء، فهي تمنحك الخصائص والوسائل التي تقوم بكلّ العمل مباشرة، بحيث تعزلك عن كتابة أيّ كود إضافيّ.. ولكنّ هذا لا يمنع أن نأخذ فكرةً سريعةً عن لغة التنسيق RTF هذه.

**لغة النصوص المنسّقة RTF Language:**

تتكوّن هذه اللغة ببساطة من بعض الأوامر "الرموز" التي توضّح تنسيق النصّ.. فمثلا البادئة \par تدلّ على بداية مقطع Paragraph في النصّ، والبادئة \b تدلّ على أنّ النصّ سميك Bold... وهكذا.

وكما تلاحظ، فإنّ كل أمر يبدأ بالعلامة "\".. فإذا كانت هذه العلامة نفسها موجودة في النصّ، فإنّها تمثّل بإضافة علامة زائدة "\\".

هذا ويوضع النصّ كلّه بين علامتي {}.

والآن سأريك التنسيق الذي ستحفظ به جملة "بسم الله الرحمن الرحيم" في هذه اللغة، لو أردنا حفظها في ملفّ .RTF:

{\rtf1\fbidis\ansi\ansicpg1256\deff0\deflang3073{\fonttbl{\f0\fswiss\fcharset178{\\*\fname Arial;}Arial (Arabic);}}

\viewkind4\uc1\pard\rtlpar\qr\lang1025\f0\rtlch\fs20\'c8\'d3\'e3 \'c7\'e1\'e1\'e5 \'c7\'e1\'d1\'cd\'e3\'e4 \'c7\'e1\'d1\'cd\'ed\'e3\lang3073\par

}

ستتساءل طبعا: ولكن أين هي حروف جملة "بسم الله الرحمن الرحيم"؟

إنّ الحروف لا تخزّن كما هي، وإنّما يتمّ كتابة الأعداد المناظرة لها في ترميز ASCII، معبرا عنها بالنظام السداسي عشر!.. هذه الحروف هي:

'c8\'d3\'e3 \'c7\'e1\'e1\'e5 \'c7\'e1\'d1\'cd\'e3\'e4 \'c7\'e1\'d1\'cd\'ed\'e3

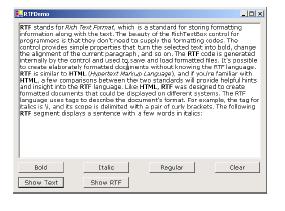
حيث ستلاحظ أنّ المسافة تمثّل بالعدد السداسي عشر c7، واللام تمثّل بالعدد e1.. وهكذا..

كما تلاحظ أنّ العدد الدالّ على الحرف تسبقه البادئة (\').

وتكفينا هذه النبذة عن لغة التنسيق RTF فنحن لن نستخدمها مباشرةً، وإنّما سنستخدم خصائص ووسائل مربّع النصّ المنسّق (وإن كانت بعض المميزات لا يمكن تمثيلها بواسطة خصائص مربّع النصّ.. هذا يعني لجوءك لهذه اللغة.. لحسن حظّك أنّ هذا نادر الحدوث).

**مشروع RTFDemo:**

تعالَ نتعرّف على مربّع النصّ المنسّق بطريقة عمليّة.. ابدأ مشروعا جديدا وسمّه RTFDemo.. ضع على النموذج نسخة من الأداة RichTextBox، وغيّر حجمها حتّى تملأ معظم مساحة النموذج.



والآن أضف زرّا للنموذج.. عند ضغط هذا الزرّ سنحوّل النصّ المحدّد في مربّع النصّ المنسّق إلى نصّ سميك Bold.. هذا هو الكود الذي يفعل ذلك:

**Private Sub bttnBold\_Click(ByVal sender As System.Object, \_**

**ByVal e As System.EventArgs) Handles bttnBold.Click**

**Dim fnt As New Font(RichTextBox1.SelectionFont, FontStyle.Bold)**

**RichTextBox1.SelectionFont = fnt**

**End Sub**

لقد ذكرنا من قبل أنّ كائن الخطّ لا يسمح بتغيير خصائصه بعد تكوينه.. لهذا لجأنا إلى تكوين خطّ جديد له نفس خطّ ومواصفات النصّ المحدّد (باستخدام خاصيّة SelectionFont)، وطرازه سميك Bold.. بعد هذا جعلنا هذا الخطّ هو خطّ النصّ المحدّد.

أضف زرا آخر، يحوّل النصّ المحدّد إلى خطّ مائل Italic، كالتالي:

Private Sub bttnItalic\_Click(ByVal sender As System.Object, \_

ByVal e As System.EventArgs) Handles bttnItalic.Click

Dim fnt As New Font(RichTextBox1.SelectionFont, FontStyle.Italic)

RichTextBox1.SelectionFont = fnt

End Sub

لاحظ أنّك لو حددت أيّ نصّ وضغطت الزر الأوّل فسيصبح سميكا.. ولو ضغطت الزر الثاني فسيصبح النصّ مائلا لكنّه لن يظلّ سميكا.. ولإصلاح هذه العيب، يجب المحافظة على التنسيقات السابقة للخطّ، مع تعديل الخاصيّة المطلوبة فحسب.. ولفعل هذا استخدم جملة كالتالية:

Dim fnt As New Font(RichTextBox1.SelectionFont, \_

RichTextBox1.SelectionFont.Style Or FontStyle.Bold)

حيث استخدمنا المعامل Or لندمج صفات الخطّ السابقة مع صفة السمك، وبهذا لن يتغيّر في الخطّ سوى أنّه سيصير سميكا.. ويمكنك أن تفعل المثل في حالة ميلان الخط.

ويحتوي تطبيق RTFDemo في مجلّد برامج هذا الفصل على المزيد من الأزرار.. فمثلا زر "Show Text" يعرض لك رسالة تعرض لك النصّ الموجود في مربّع النص(باستخدام خاصيّة Text)، وزرّ "Show RTF" يعرض لك تنسيق RTF للنصّ الموجود في مربّع النصّ (باستخدام خاصيّة RTF).. جرّب هذه الإمكانيّات.

**خصائص مربّع النصّ المنسّق:**

كثير من خصائص هذه الأداة مماثلة لخصائص مربّع النصّ التقليديّ، لذلك فلن نتحدّث إلا عن الخصائص الجديدة.

وفي الجدول التالي الخصائص التي تتعامل مع النصّ المحدّد في مربّع النصّ المنسّق.. هذه الخصائص قابلة للقراءة وتغيير قيمتها:

|  |  |
| --- | --- |
| SelectedText | النصّ المحدّد.. وهنا نتعامل مع الحروف فقط، ولا دخل لنا بالتنسيق.. والجملة التالية تحوّل حروف النصّ إلى حروف كبيرة Capital:  RTB1.SelectedText=UCase(RTB1.SelectedText) |
| SelectedRTF | تنسيق النصّ المحدّد.. هذا التنسيق مكتوب بلغة RTF، وهو يتضمّن حروف النصّ بداخله.. ويمكنك الاستفادة من هذه الخاصيّة في نقل نصّ بتنسيقه من مربّع نصّ منسّق إلى آخر. |
| SelectionStart | موضع بداية التحديد.. تذكّر أنّ الموضع صفر هو موضع تحديد أوّل حرف في مربّع النصّ. |
| SelectionLength | طول التحديد (عدد الحروف المحدّدة). |
| SelectionFont | خطّ النصّ المحدّد.. ونعني بالخطّ هنا كائن الخطّ، وهو يحتوي على كلّ خصائص الخط (اسمه، وحجمه، وسماته). |
| SelectionColor | لون النصّ المحدّد. |
| SelectionAlignment | محاذاة السطر الذي يوجد به النصّ المحدّد (لليمين، لليسار، للوسط).. طبعا ستحتاج هذه الخاصيّة هي وخاصية RightToLeft مع اللغة العربيّة (راجع [الفصل السادس](file:///C:\Users\Administrator\Desktop\01-%20أساسيّات%20اللغة\06-%20Basic%20Controls%20الأدوات%20الأساسيّة\1-%20مربّع%20النصّ\6-%20تغيير%20لغة%20لوحة%20المفاتيح.htm) لمعرفة كيفيّة تغيير لغة لوحة المفاتيح برمجيّا).  ولو حدثت معك أيّ مشاكل في تهيئة مربّع النصّ المنسّق للكتابة باللغة العربيّة عند تشغيل التطبيق، فيمكنك أن تكتب ملفّ في Word، وتنسّقه تنسيقا عربيّا، واحفظه بتنسيق RTF، واحتفظ به في مجلّد البرنامج، وعند بدء التشغيل، قم بتحميل هذا الملفّ في خاصيّة Rtf الخاص بمربّع النصّ المنسّق، وبهذا تحلّ المشكلة (لو كان هناك أيّ نصّ مكتوب في ملفّ Word، فامحُه من خاصيّة Text) |
| SelectionProtected | لحماية النصّ المحدّد.. إذا جعلت هذه الخاصيّة True، فإنّ المستخدم لن يستطيع تغيير أو حذف المقطع المحدّد (حتّى بعد إزالة التحديد). |
| SelectionBullet | لوضع علامة تعداد نقطي.. فإذا جعلت هذه الخاصيّة True فستوضع العلامة التالية عند بداية المقطع:   * (أي كلمات في المقطع). |
| SelectionIndent | هامش بداية السطر الأوّل في مقطع Paragraph النصّ المحدّد. |
| SelectionHangingIndent | هامش كلّ مقطع النصّ المحدّد ما عدا السطر الأوّل (بتعبير Word: مسافة بادئة معلّقة). |
| SelectionRightIndent | هامش نهاية كلّ سطور المقطع. |

**ملحوظتان:**

1- ما يختصّ من الخصائص السابقة بالتنسيق، يُرجع Nothing لو كان الجزء المحّدد من النصّ يحتوى على تنسيقات مختلفة.. فمثلا سترجع خاصيّة SelectionColor القيمة Nothing لو كان النصّ المحدّد يحتوى على لونين مختلفين.. لهذا كن على حذر عند استخدام هذه الخصائص، واختبر قيمتها قبل أن تبني عليها أيّ خطوات (استخدم تعبير IsNothing)، حتّى لا يحدث أيّ خطإ أو أيّة نتيجة غير مرغوبة.

2- لتغيير سمات أيّ مقطع من النصّ، يجب تحديده أولا.. والمثال التالي يريك كيف تغيّر لون جزء من النصّ إلى اللون الأحمر:

**RichTextBox1.SelectionStart = 3**

**RichTextBox1.SelectionLength = 10**

**RichTextBox1.SelectionColor = Color.Red**

**RichTextBox1.SelectionLength = 0**

لاحظ أنّنا أعدنا طول النصّ المحدّد إلى صفر بعد تلوين الجزء المطلوب.. إنّ هذا ليس إجباريّا، ولكنّك قد تريده في بعض الحالات، ذلك حتّى لا يظلّ النصّ محدّدا بعد تغيير لونه.

**تدريب:**

تحتاج وأنت تغيّر تنسيق بعض الجمل، ألا يرى المستخدم عمليّة التحديد وهي تتمّ ثمّ تزال في مربّع النصّ المنسّق (يعني: منظر غير جميل).

افترض أنّك تقوم بتحميل وثيقة في مربّع النصّ.. هذه الوثيقة بها بعض الكلمات الهامّة التي يجب تلوينها باللون الأحمر (يمكن أن تكون هذه الكلمات مميّزة بوجود رمز يسبقها، مثل @ أو $ أو أيّ رمز آخر.. وطبعا عليك ألا تعرض هذا الرمز في مربّع النصّ).. في هذه الحالة سيرى المستخدم التحديد وهو يوضع ويزال أكثر من مرّة، والمنزلق الرأسيّ وهو يتحرّك مع تغيّر موضع التحديد في كلّ مرّة (لو كان النصّ طويلا ممّا يستلزم وجود منزلق رأسيّ).. إنّ هذا ليس مظهرا طيّبا لتطبيقك، والأفضل أن تخفيه عن المستخدم.. ولفعل ذلك لديك حلان:

أ. أن تخفي مربّع النصّ المنسّق كلّه بمنح خاصيّة Visible القيمة False، حتّى إذا أتممت العمليّة أعدت قيمتها إلى True.. وفي الغالب لن يلاحظ المستخدم اختفاء مربّع النصّ وظهوره، لأنّ انشغال VB في تنفيذ العمليّة سيمنعه من إنعاش النموذج.. وبهذا لن يرى المستخدم عمليّات التحديد وهي تتمّ.

ب. أن تعرّف متغيّرا من خليّة مربّع النصّ المنسّق كالتالي:

**Dim Rt As New RichTextBox()**

الآن يمكنك التعامل مع المتغيّر Rt كأنّك تتعامل مع أيّ مربّع نصّ منسّق، مع فارق وحيد، هو أنّه ليس مرسوما على النموذج.. هنا يمكنك أن تحمّل الوثيقة في المتغيّر Rt، وتلوّن الكلمات المميّزة بها، وذلك باستخدام نفس الخصائص التي تعلّمناها.. وبعد أن تنتهي، يمكنك أن تنقل النصّ المنسّق من المتغيّر Rt إلى مربّع النصّ المنسّق بالجملة التالية:

**RichTextBox1.Rtf = Rt.Rtf**

وبهذا ستظهر الوثيقة ملوّنة الكلمات في مربّع النصّ المنسّق دونَ أن يلاحظ المستخدم أيّ شيء غير طبيعيّ.. وأنا أنصحك باستخدام هذا الحلّ أفضل من الحلّ الأوّل.

والآن: اكتب تطبيقا به قائمة ملفّ File Menu، بها الأمر فتح، الذي يعرض للمستخدم مربّع حوار فتح ملفّ، يسمح فقط باختيار الملفّات النّصّيّة .Txt، حيث يُحمّل في مربّع النصّ المنسّق، الملفّ النصّي الذي يختاره المستخدم (يمكنك استخدام وسائل فتح وقراءة الملفّات التي درسناها في فصل الملفّات، أو يمكنك استخدام الوسيلة LoadFile التي سنتعرّف عليها بعد قليل).. ولو كانت هناك كلمات تبدأ بالرمز @، فقم بتلوينها باللون الأحمر واحذف هذا الرمز من مربّع النصّ المنسّق.. طبعا دون أن يلاحظ المستخدم هذه العمليّة.. وأعتقد أنّك تعرف كيف تبحث عن حرف في النصّ (يمكنك كذلك استخدام الوسيلة Find التي سنتعرّف عليها بعد قليل).. هذا مع ملاحظة أنّ الكلمة تنتهي بمسافة أو بنهاية السطر vbCr أو بنهاية النصّ كلّه، أو بإحدى علامات الترقيم التالية:

. : ؟ ! ، " ـ ( ) [ ] { }

وحبّذا لو قمت كذلك بتلوين علامات الترقيم باللون الأزرق.

ولاختبار هذا التطبيق، أنشئ بعض ملفّات النصوص، واكتب بها ما يروقك، وضع العلامة @ في بداية بعض الكلمات، ولا تنسَ وضع بعض علامات الترقيم، ثمّ افتح هذه الملفّات في تطبيقك.

حاول كذلك أن تضيف هذه الإمكانيّة للتطبيق: عندما يكتب المستخدم أيّ علامة ترقيم في مربّع النصّ المنسّق تظهر باللون الأزرق، وأيّ حرف آخر يظهر باللون الأسود.. والفكرة بسيطة: استخدم الحدث KeyPress لاختبار الحرف قبل أن يكتب، فإذا وجدت أنّه علامة ترقيم، فغيّر خاصيّة SelectionColor إلى اللون الأزرق.. إنّ هذا سيؤثّر على الحرف الذي سيكتب.. فإذا لم يكن الحرف من علامات الترقيم، فغيّر خاصيّة SelectionColor إلى اللون الأسود.

عموما ستجد هذا التطبيق في مجلّد برامج هذا الفصل، باسم: ColoringWords.. حذار: لا تغشّ من هذا التطبيق إلا إذا أعلنتَ فشلَك تماما في كتابة هذا المثال البسيط!.. (لا تنسَ أننّا الآن في الفصول المتقدّمة، والمفترض أنّك قد صرت على قدرٍ لا بأس به من الاحتراف.. أو على الأقل: الإجادة!).

**ومن خصائص مربّع النصّ المنسّق أيضا:**

**النصّ Text، والنصّ المنسّق RTF:**

تعيد الخاصيّة الأولى النصّ بدون تنسيقه، بينما تعيد الخاصيّة الثانية كود RTF المعبّر عن النصّ وتنسيقه.

**هامش العلامة النقطيّة BulletIndent:**

إذا جعلت لخاصيّة SelectionBult القيمة True، فإنّ العلامة النقطيّة ستظهر في السطر الأوّل من المقطع المحدّد.. ولتحديد هامش العلامة النقطيّة، استخدم خاصيّة BulletIndent، حيث يمكنك أن تضع فيها القيمة التي تريد.

**تحديد الكلمة تلقائيّا AutoSelectWord:**

لو جعلت قيمة هذه الخاصيّة True، فسيتمكّن المستخدم من تحديد كلّ الكلمة بنقرها مرّتين بالفأرة.

**اسـم عمليّة التراجع UndoActionName:**

تخبرك هذه الخاصيّة باسم العمليّة التي ستتراجع عنها، سواء أكانت تراجعا عن كتابة Typing أو تراجع عن حذف Delete.. وقد تكون العمليّة بدون اسم محدّد Unknown.. ومن السمات الرائعة في مربّع النصّ المنسّق، قدرته على التراجع عن مجموعة كبيرة من الأحداث السابقة واحدا بعد الآخر، فهو ليس كمربّع النصّ العاديّ الذي لا يستطيع التراجع إلا عن آخر عمليّة فقط.

**اسم عمليّة الإعادة RedoActionName:**

تخبرك هذه الخاصيّة باسم العمليّة التي ستعيدها بعد أن تراجعت عنها.. وهي ترجع نفس القيم كما في الخاصيّة السابقة.

**تنسيق عنوان موقع إنترنت DetectURLs:**

إذا جعلت هذه الخاصيّة True، فإنّ مربّع النصّ المنسّق سيحوّل أيّ نصّ يمثّل عنوان موقع على الإنترنت (مثلا له الصيغة WWW.Name.COM) إلى وصلة فائقة Hyperlink، بحيث سيبدو في مربّع النصّ المنسّق كالتالي:

[WWW.Name.COM](http://www.name.com/)

وعندما يمرّ المستخدم بمؤشّر الفأرة فوق هذه الوصلة، سيتغيّر شكل مؤشّر الفأرة إلى يد تشير بإصبع.. ولو ضغط المستخدم هذه الوصلة، فسيتمّ إطلاق الحدث "ضغط الوصلة" LinkClicked، وبهذا يمكنك عرض الصفحة المناظرة لهذه الوصلة، بكتابة الكود المناسب في هذا الحدث، كالتالي:

**Sub RichTextBox1\_LinkClicked(sender As Object, e As \_**

**System.Windows.Forms.LinkClickedEventArgs) \_**

**Handles RichTextBox1.LinkClicked**

**System.Diagnostics.Process.Start(e.LinkText)**

**End Sub**

حيث إنّ الخليّة System.Diagnostics.Process تمنحك الوسيلة Start، التي تشغّل أيّ برنامج.

**ملحوظة:**

رغم التطوّرات الكبيرة التي لحقت بمربّع النصّ المنسّق في هذا الإصدار، إلا إنّه ما زال لا يستطيع أن يعرض النصّ مضبوطا (بمحاذاته من اليمين واليسار في نفس الوقت) كما يفعل Word.. فإذا كان هذا الأمر مهمّا جدّا بالنسبة لك، فعليك أن تنسى مربّع النصّ المنسّق وتستخدم الأداة WebBrowser، وهي تعتبر نسخة من متصفّح الإنترنت Internet Explorer، حيث يمكنك أن تعرض بها ملفّات HTML وهي تمنحك خاصيّة الضبط وقدرات أخرى عديدة (يمكنك أن تنشئ ملفاتك في Word ثمّ تحفظها بتنسيق HTML أو الأفضل بتنسيق Web Page Filtered للتخلّص من أيّ معلومات إضافيّة تخصّ Word).. ولكنّ هذا لن يفيدك إلا في حالة واحدة فقط: إذا كنت ستعرض للمستخدم معلومات غير قابلة للتغيير (مثل ملفات المساعدة).. وستجد بعض المعلومات عن هذه الأداة في [نهاية الفصل بعد القادم](file:///C:\Users\Administrator\Desktop\vb.net\13-%20بناء%20الأدوات%20الخاصّة\5-%20الأدوات%20من%20النوع.htm).

**وسائل مربّع النصّ المنسّق:**

**حفظ لملفّ SaveFile:**

استخدم هذه الوسيلة لحفظ محتويات مربّع النصّ المنسّق في ملفّ.. ولهذه الوسيلة الصيغة التالية:

**RichTextBox1.SaveFile(مسار الملفّ, نوع الملفّ)**

ويمكن ألا ترسل المعامل الثاني، حيث سيتمّ حفظ الملفّ بامتداد .rtf للمحافظة على تنسيقه.. كما يمكن أن ترسل المعامل الثاني بإحدى قيم المرقّم RichTextBoxStreamType، وهي:

|  |  |
| --- | --- |
| PlainText | تحفظ حروف النصّ بغير تنسيق. |
| RichNoOLEObjs | تحفظ النصّ بدون تنسيقه، ودون أن تحفظ أيّ كائنات OLE موجودة بداخل مربّع النصّ كالصور مثلا. |
| RichText | تحفظّ النصّ بتنسيقه. |
| TextTextOLEObjs | تحفظ النصّ بدون تنسيق، ولكن تحفظ كائنات OLE. |
| UnicodePlainText | تحفظ النصّ بتنسيق Unicode. |

**تحميل ملفّ LoadFile:**

تحمّل محتويات ملفّ لوضعها في مربّع النصّ المنسّق.. وصيغة هذه الدالة مماثلة لصيغة الدالة السابقة:

**RichTextBox1.LoadFile(مسار الملفّ, نوع الملفّ)**

حيث إنّ نوع الملفّ هو معامل اختياريّ، وقيمه مماثلة لقيم نظيره في وسيلة الحفظ.

**نسخ Copy، قصّ Cut، لصق Paste:**

يمنحك مربّع النصّ المنسّق هذه الوسائل للقيام بعمليات النسخ والقصّ واللصق.. والوسيلتان الأوليان لا تأخذان أيّ معاملات، أمّا وسيلة اللصق فتأخذ معاملا، هو نوع تنسيق البيانات التي ستلصقها، حيث إنّ هذه البيانات قادمة من لوحة القصاصات Clipboard.. ويمكنك استخدام الوسيلة "يمكن اللصق" **CanPaste** لتعرف إذا كان نوع البيانات الموجودة في لوحة القصاصات يمكن وضعه في مربّع النصّ المنسّق أم لا.

والمثال التالي يلصق محتويات لوحة القصاصات لمربّع النصّ المنسّق إذا كانت نصوصا:

**If Clipboard.GetDataObject.GetDataPresent(DataFormats.Text) Then**

**RichTextBox.Paste(DataFormats.Text)**

**End If**

لاحظ كذلك أن وسيلة اللصق، هي الطريقة التي تتمكّن بها من وضع (لصق) الصور في الموضع الحاليّ من مربّع النصّ المنسّق.. والدالة التالية تستقبل منك اسم ملف الصورة، لتدرجها في الموضع الحاليّ من مربّع النصّ:

**Private Function PasteMyBitmap(ByVal Filename As String) As Boolean**

**'Open an bitmap from file and copy it to the clipboard.**

**Dim MyBitmap As Bitmap**

**MyBitmap = Bitmap.FromFile(Filename)**

**' نسخ الصرة للوحة القصاصات**

**Clipboard.SetDataObject(MyBitmap)**

**Dim MyFormat As DataFormats.Format**

**MyFormat = DataFormats.GetFormat(DataFormats.Bitmap)**

**' محاولة لصق الصورة بعد التأكّد من تنسيقها**

**If RichTextBox1.CanPaste(MyFormat) Then**

**RichTextBox1.Paste(MyFormat)**

**PasteMyBitmap = True**

**Else**

**MessageBox.Show("هذه البيانات لا يمكن لصقها")**

**PasteMyBitmap = False**

**End If**

**End Function**

وعليك أن تنتبه إلى ملائمة حجم الصورة مع أبعاد مربّع النصّ المنسّق.. راجع الفصل الخاصّ بالرسم والتلوين من أجل مزيد من التفاصيل.

**بحث Find:**

هذه الوسيلة تشبه الوسيلة InStr.. وهي تقوم بالبحث في نصّ مربّع النصّ المنسّق عن النصّ الذي تريده، وذلك في صورتها البسيطة التالية:

**RichTextBox1.Find(نصّ)**

وكما عهدت، لو كان النصّ موجودا تعيد هذه الوسيلة موضع أوّل حرف منه مع تحديد نصّ البحث في مربّع النصّ، وإن لم يكن موجودا فإنّها تعيد -1.

وهناك صيغة أخرى كالتالي:

**RichTextBox1.Find(نصّ, طراز البحث)**

حيث يأخذ طراز البحث واحدة من قيم المرقّم RichTextBoxFinds التالية:

|  |  |
| --- | --- |
| MatchCase | البحث مع مراعاة حالة الأحرف. |
| NoHighlight | عند العثور على النصّ لا يتمّ تحديده في مربّع النصّ. |
| Reverse | يبدأ البحث من نهاية مربّع النصّ لبدايته. |
| WholeWord | البحث عن النصّ بحيث يكون كلمة كاملة مستقلّة، وليس جزءا من كلمة أخرى في مربّع النصّ. |
| None | لا يوجد طراز بحث خاصّ. |

ويمكنك أن تدمج أكثر من قيمة من هذه القيم معا باستخدام عمليّة Or.. انظر المثال التالي:

**Dim Pos As Integer**

**Pos = RichTextBox1.Find("كلمة", RichTextBoxFinds.WholeWord \_**

**Or RichTextBoxFinds.Reverse)**

كما توجد صيغتان أخريان لهذه الوسيلة، تسمحان لك بتحديد موضع بداية البحث في مربّع النصّ المنسّق، أو موضع نهاية البحث، كالتالي:

**Pos = RichTextBox1.Find(نصّ, موضع بداية البحث, طراز البحث)**

**Pos = RichTextBox1.Find(نصّ, موضع بداية البحث, موضع نهاية البحث, طراز البحث)**

وتوجد صيغ أخرى لهذا الوسيلة، تأخذ مصفوفة حروف، اتبحث عنها، حيث تعيد موضع أوّل حرف تجده من هذه الحروف.. وللأسف، لا تقوم الوسيلة في هذه الحالة بتحديد الحرف الذي تمّ العثور عليه:

**Dim Ch As Char = {"/" , "@", "#"}**

**Pos = RichTextBox1.Find(Ch)**

**Pos = RichTextBox1.Find(Ch, موضع بداية البحث)**

**Pos = RichTextBox1.Find(Ch, موضع بداية البحث, موضع نهاية البحث)**

**معرفة الحرف من الإحداثيّات GetCharFromPosition:**

استخدم هذه الوسيلة لتعرف الحرف الموجود في مربّع النصّ المنسّق في إحداثيّات معيّنة أو قريبا منها.. ويمكنك استخدام هذه الخاصيّة لمعرفة أقرب حرف للموضع الذي يمرّ فوقه مؤشّر الفأرة، وذلك بكتابة الجملة التالية في حدث حركة الفأرة MouseMove:

**Private Sub RichTextBox1\_MouseMove(sender As Object, e \_**

**As System.Windows.Forms.MouseEventArgs) \_**

**Handles RichTextBox1.MouseMove**

**Dim P As New Point(e.X, e.Y)**

**' اعرض أقرب حرف لمؤشّر الفأرة كعنوان للنموذج**

**Me.Text = RichTextBox1.GetCharFromPosition(p)**

**End Sub**

**معرفة موضع الحرف من الإحداثيّات GetCharIndexFromPosition:**

نظرا لأنّ حروف النصّ يمكن أن تتكرّر، فقد لا تجدُ الدالةَ السابقة مفيدةً في تحديد Select موضع أقرب حرف لإحداثيات مؤشّر الفأرة، لهذا فإنّ لديك هذه الوسيلة التي تعيد لك رقم الحرف القريب من مؤشّر الفأرة في مربّع النصّ.. والكود التالي يعرض **الكلمة** التي يمرّ فوقها مؤشّر الفأرة كعنوان للنموذج:

**Private Sub RichTextBox1\_MouseMove(sender As Object, e \_**

**As System.Windows.Forms.MouseEventArgs) \_**

**Handles RichTextBox1.MouseMove**

**If RichTextBox1.TextLength = 0 Then Exit Sub**

**Dim P As New Point(e.X, e.Y)**

**Dim Pos, StartPos, EndPos As Integer**

**Dim C As Char**

**' لو كان الحرف الحالي مسافة أو حرف بداية سطر أو حرف نهاية سطر، غادر الإجراء**

**C = RichTextBox1.GetCharFromPosition(P)**

**Select Case C**

**Case " "c, Chr(10), Chr(13)**

**Exit Sub**

**End Select**

**Pos = RichTextBox1.GetCharIndexFromPosition(P)**

**' تحديد موضع بداية الكلمة**

**If Pos > 0 Then**

**' موضع بداية الكلمة هو موضع أقرب حرف (سواء المسافة أو بداية السطر)**

**' للموضع الحالي**

**StartPos = Math.Max(\_**

**RichTextBox1.Text.LastIndexOf(" ", Pos - 1), \_**

**RichTextBox1.Text.LastIndexOf(Chr(10), Pos - 1))**

**If StartPos = -1 Then**

**' هذه أوّل كلمة في السطر الأوّل، لهذا لا تسبقها مسافات أو بداية سطر**

**StartPos = 0**

**Else**

**' يجب أن نضيف 1 على موضع البداية،**

**' لأنّ المسافة أو بداية السطر ليستا ضمن الكلمة**

**StartPos += 1**

**End If**

**' Else**

**' لو كانت الفأرة تشير إلى أوّل حرف، فسيكون موضع بداية الكلمة هو صفر**

**' إنّ القيمة الابتدائيّة للمتغيّرات الرقميّة هي صفر، لذلك لن نكتب جملة تفعل ذلك**

**'StartPos = 0**

**End If**

**' تحديد موضع نهاية الكلمة**

**If Pos < RichTextBox1.TextLength - 1 Then**

**' تذكّر أنّ موضع أوّل حرف في النصّ هو 0، وآخر حرف هو طول النصّ -1**

**Dim I, J As Integer**

**I = RichTextBox1.Find(" ", \_**

**Pos + 1, RichTextBoxFinds.NoHighlight)**

**J = RichTextBox1.Find(Chr(13), \_**

**Pos + 1, RichTextBoxFinds.NoHighlight)**

**' نهاية الكلمة عند أقرب موضع للحرف التالي توجد به مسافة أو نهاية سطر**

**EndPos = Math.Min(I, J)**

**' لو كان أحد المتغيّرين في السطر السابق -1**

**' نتيجة لعدم وجود مسافة أو نهاية سطر، فيجب أن نأخذ الرقم الآخر،**

**' الذي سيكون في هذه الحالة هو أكبر الرقمين**

**If EndPos = -1 Then EndPos = Math.Max(I, J)**

**' إذا كان الناتج ما يزال -1، فهذا يعني أنّ كلا الرقمين كان -1**

**If EndPos = -1 Then**

**' الكلمة هي آخر كلمة في السطر الأخير**

**EndPos = RichTextBox1.TextLength - 1**

**Else**

**' يجب طرح 1 لأنّ المسافة ونهاية السطر ليستا ضمن الكلمة**

**EndPos -= 1**

**End If**

**Else**

**' مؤشّر الفأرة يشير لآخر حرف في الكلمة**

**EndPos = Pos**

**End If**

**' لا تنس أنّ طول الكلمة = النهاية – البداية + 1**

**Me.Text = RichTextBox1.Text.Substring(StartPos, \_**

**EndPos - StartPos + 1)**

**End Sub**

ويمكن استخدام وسيلتي خليتي النصوص IndexOfAny و LastIndexOfAny لتبسيط الإجراء السابق، حيثّ إنّ كلا منهما تبحث عن مجموعة من الحروف، بحيث تعيد موضع أوّل ما تعثر عليه منها.. إنّ هذا يمكّنك من اعتبار علامات الترقيم كفواصل للكلمات، وليس المسافة فحسب.. انظر للكود التالي:

**Private Sub RichTextBox1\_MouseMove(sender As Object, e \_**

**As System.Windows.Forms.MouseEventArgs) \_**

**Handles RichTextBox1.MouseMove**

**If RichTextBox1.TextLength = 0 Then Exit Sub**

**Dim P As New Point(e.X, e.Y)**

**Dim Pos, StartPos, EndPos As Integer**

**Dim C As Char**

**C = RichTextBox1.GetCharFromPosition(P)**

**Select Case C**

**Case " "c, Chr(10), Chr(13)**

**Exit Sub**

**End Select**

**' مصفوفة البحث، ستحتوى على كلّ فواصل الكلمات**

**Dim X() As Char = {" "c, "("c, ")"c, "["c, "]"c, "{"c, "}"c, \_**

**"\_"c, ","c, "،"c, ":"c, "."c, "/"c, "\"c ,"!"c ,"?"c ,\_**

**"؟"c, Chr(10)}**

**Pos = RichTextBox1.GetCharIndexFromPosition(P)**

**If Pos > 0 Then**

**StartPos = RichTextBox1.Text.LastIndexOfAny(X, Pos - 1)**

**If StartPos = -1 Then**

**StartPos = 0**

**Else**

**StartPos += 1**

**End If**

**End If**

**If Pos < RichTextBox1.TextLength - 1 Then**

**EndPos = RichTextBox1.Text.IndexOfAny(X, Pos + 1)**

**If EndPos = -1 Then**

**EndPos = RichTextBox1.TextLength - 1**

**Else**

**EndPos -= 1**

**End If**

**Else**

**EndPos = Pos**

**End If**

**Me.Text = RichTextBox1.Text.Substring(StartPos, \_**

**EndPos - StartPos + 1)**

**End Sub**

**ملحوظة:**

لسبب لا أدريه ـ لا تستطيع الوسيلة IndexOfAny ولا الوسيلة LastIndexOfAny أن تعثر في النصّ المنسّق على الحرف Chr(13) الذي يعبّر عن نهاية السطر، ويبدو أن خاصية Text الخاصة بمربّع النصّ المنسّق لا تحتوى على هذا الحرف، وتكتفي بالحرف Chr(10) كفاصل للأسطر.. لهذا فإنّ الكود السابق يبحث عن الحرف Chr(10) الذي يمثّل بداية السطر الحالي ونهاية السطر السابق!.. هذا على خلاف الوسيلة Find التي استطاعت العثور على كل من الحرفين Chr(13) و Chr(10) بطريقة طبيعيّة.. انتبه جيّدا لهذا الفارق، فسيؤثّر على نتائج برنامجك.. ونتيجة لذلك، لو حاولت أن تنسخ نصّا يتكوّن من عدّة أسطر من مربّع النصّ المنسّق إلى مربّع نصّ عاديّ بجملة كالتالية:

**TextBox1.Text = RichTextBox1.Text**

فإنّ النص سيبدو في مربّع النصّ العاديّ كأنّه سطر واحد فقط!!.. ولحلّ هذه المشكلة، استبدل كلّ حرف Chr(10) (أو الثابت الذي يمثّله وهو vbLf) بالحرفين Chr(13) + Chr(10) (أو بالثابت الذي يعادلهما وهو vbCrLf)، قبل وضع النصّ في مربّع النصّ، كالتالي:

**TextBox1.Text = RichTextBox1.Text.Replace(vbLf, vbCrLf)**

أو يمكنك نسخ النصّ من مربّع النصّ المنسّق إلى لوحة القصاصات، ثمّ لصقه في مربّع النصّ العاديّ.

وعليك أن تتأكّد في كلّ إصدار جديد، أنّ ميكروسوفت لم تحلّ هذه المشكلة.

**أحداث مربّع النصّ المنسّق:**

يمتلك مربّع النصّ المنسّق نفس الأحداث المعروفة.. وسنكتفي هنا بالإشارة إلى الأحداث الجديدة:

**انزلاق أفقي HScroll:**

يتمّ إطلاق هذا الحدث عندما يحرّك المستخدم المنزلق الأفقيّ.

**انزلاق رأسيّ VScroll:**

يتمّ إطلاق هذا الحدث عندما يحرّك المستخدم المنزلق الرأسيّ.

**ضغط الوصلة LinkClicked:**

ذكرنا من قبل أنّ هذا الحدث ينطلق عندما يضغط المستخدم نصّا يمثّل وصلة فائقة HyperLink في مربّع النصّ المنسّق.

**محميّ Protected:**

ينطلق إذا حاول المستخدم تغيير النصّ المحميّ في مربّع النصّ المنسّق (استخدمه لإطلاق رسالة تحذير للمستخدم).

**تغيّر التحديد SelectionChanged:**

يحدث إذا غيّر المستخدم بداية التحديد أو طوله أو الموضع الحالي لمؤشّر الكتابة.. ويمكنك الاستفادة من هذا الحدث في اختبار تنسيق الموضع الحالي من النصّ لعرضه في القوائم.. فمثلا لو لديك قائمة عند ضغطها تحوّل النصّ المحدّد إلى نصّ سميك Bold مع وضع علامة (✓) على القائمة (بجعل خاصيّة Checked بالقيمة True)، بحيث لو ضغطتها مرّة أخرى تعيد النصّ إلى نصّ غير سميك، مع إزالة علامة (✓).. في هذه الحالة لو حدّد المستخدم أيّ نصّ، يجب أن تعرف إذا كان سميكا أم لا، حتّى يتمّ وضع قيمة خاصيّة Checked للقائمة.. إنّ ذلك يتمّ في حدث تغيير التحديد كالتالي:

**Private Sub RichTextBox1\_SelectionChanged(sender As Object, e As \_**

**System.EventArgs) Handles RichTextBox1.SelectionChanged**

**If Not RichTextBox1.SelectionFont Is Nothing Then**

**MenuBold.Checked = RichTextBox1.SelectionFont.Bold**

**Else**

**MenuBold.Checked = False**

**End If**

**End Sub**

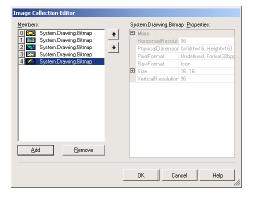
ولديك في مجلّد برامج هذا الفصل التطبيق RTFPad، وهو يمثّل محرّر نصّ يشابه تطبيق WpordPad الموجود مع الويندوز (يمكنك أن تجده في قائمة Programs تحت المجّلد Accessories).. وفي هذا التطبيق سترى كيف يمكنك التعامل مع مربّع النصّ المنسّق بمعظم إمكانيّاته، بما فيها اختيار هوامش المقاطع، وألوان وخطوط أيّ أجزاء من النصّ.. وقتا ممتعا!

**قائمة الصور ImageList Control:**

سرعان ما ستكتشف أنّ هذه الأداة أبسط من أن نشرحها في فصل يحمل عنوان "أدوات متقدّمة"، فكلّ ما تفعله هو أنّها تخزن مجموعة من الصور وتحمّلها في الذاكرة لتسريع استخدامها في البرنامج!.. إذن لماذا؟ ـ أسمعك تتساءل!

الموضوع وما فيه أنّ هذه الأداة حيويّة جدا وتستخدم كثيرا مع الأداتين اللتين سنشرحهما بعدها.

ضع نسخةً من هذه الأداة على النموذج، وفي نافذة الخصائص اختر خاصيّة "الصور" Images، واضغط زر الانتقال ليظهر لك نافذة محرّر مجموعة الصور، التي من خلالها يمكنك أن تحمّل كلّ ما تريد من الصور، باختيار ملفّاتها من على القرص الصلب، وذلك بضغط زر Add، ليظهر لك مربّع حوار "فتح ملفّ"، ومن خلاله تختار ملفّات الصور، حيث ستأخذ كلّ صورة رقما تبعا لترتيب تحميلها.



وكما اتفقنا من قبل، فإنّ أيّ صورة يتمّ تحميلها في وقت التصميم يتمّ حفظها في ملفّات البرنامج، بحيث لن تحتاج لتوزيع ملفّاتها الأصليّة مع البرنامج، ولن تقلق من تغيّر مسار الصور على جهازك.

ومن خلال نافذة الخصائص أيضا، يمكنك اختيار قيمة لون الشفافية TransparentColor، ليصير لون الشفافية لكلّ الصور الموجودة في قائمة الصور، حيث لن يظهر هذا اللون عند رسم الصورة (لو كان موجودا بها طبعا)، وإنّما سيشفّ عمّا تحته.

ويمكن إضافة الصور من خلال الكود، عن طريق استخدام الوسيلة Add الخاصّة بمجموعة الصور Images collection، كالتالي:

**ImageList1.Images.Add(Image.FromFile(مسار الصورة))**

حيث إنّ كلّ عنصر في مجموعة الصور Images هو كائن صورة Image.

وسنرى لاحقا كيف يمكن استخدام هذه الأداة مع الشجرة TreeView وقائمة العرض ListView.

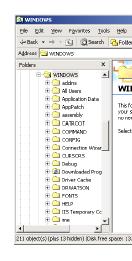
**أداة العرض المتفرّع "الشجرة" TreeView Control**

* [**ما هي الشجرة؟:**](file:///C:\Users\Administrator\Desktop\vb.net\d-%20%20مواضيع%20متقدّمة\11-%20أدوات%20متقدّمة\3-%20الشجرة\2-%20ما%20هي%20الشجرة.htm)
* [**خصائص الشجرة:**](file:///C:\Users\Administrator\Desktop\vb.net\d-%20%20مواضيع%20متقدّمة\11-%20أدوات%20متقدّمة\3-%20الشجرة\3-%20خصائص%20الشجرة.htm)
* [**بعض وسائل الشجرة:**](file:///C:\Users\Administrator\Desktop\vb.net\d-%20%20مواضيع%20متقدّمة\11-%20أدوات%20متقدّمة\3-%20الشجرة\4-%20بعض%20وسائل%20الشجرة.htm)
* [**إضافة العناصر للشجرة في وقت التصميم:**](file:///C:\Users\Administrator\Desktop\vb.net\d-%20%20مواضيع%20متقدّمة\11-%20أدوات%20متقدّمة\3-%20الشجرة\5-%20إضافة%20العناصر%20للشجرة%20في%20وقت%20التصميم.htm)
* [**التعامل مع العناصر من الكود:**](file:///C:\Users\Administrator\Desktop\vb.net\d-%20%20مواضيع%20متقدّمة\11-%20أدوات%20متقدّمة\3-%20الشجرة\6-%20التعامل%20مع%20العناصر%20من%20الكود.htm)
* [**فرع الشجرة TreeNode:**](file:///C:\Users\Administrator\Desktop\vb.net\d-%20%20مواضيع%20متقدّمة\11-%20أدوات%20متقدّمة\3-%20الشجرة\7-%20فرع%20الشجرة.htm)
* [**مشروع TreeViewDemo:**](file:///C:\Users\Administrator\Desktop\vb.net\d-%20%20مواضيع%20متقدّمة\11-%20أدوات%20متقدّمة\3-%20الشجرة\8-%20مشروع%20الشجرة.htm)
* [**أهمّ أحداث الشجرة:**](file:///C:\Users\Administrator\Desktop\vb.net\d-%20%20مواضيع%20متقدّمة\11-%20أدوات%20متقدّمة\3-%20الشجرة\9-%20أهمّ%20أحداث%20الشجرة.htm)
* [**المرور عبر كلّ عناصر الشجرة:**](file:///C:\Users\Administrator\Desktop\vb.net\d-%20%20مواضيع%20متقدّمة\11-%20أدوات%20متقدّمة\3-%20الشجرة\10-%20المرور%20عبر%20كلّ%20عناصر%20الشجرة.htm)
* [**البحث عن عنصر في الشجرة:**](file:///C:\Users\Administrator\Desktop\vb.net\d-%20%20مواضيع%20متقدّمة\11-%20أدوات%20متقدّمة\3-%20الشجرة\10-%20البحث%20في%20الشجرة.htm)
* [**عرض الشجرة من اليمين لليسار:**](file:///C:\Users\Administrator\Desktop\vb.net\d-%20%20مواضيع%20متقدّمة\11-%20أدوات%20متقدّمة\3-%20الشجرة\09-%20عرض%20الشجرة%20من%20اليمين%20لليسار.htm)
* [**حفظ عناصر الشجرة في ملفّ:**](file:///C:\Users\Administrator\Desktop\vb.net\d-%20%20مواضيع%20متقدّمة\11-%20أدوات%20متقدّمة\3-%20الشجرة\11-%20حفظ%20عناصر%20الشجرة%20في%20ملفّ.htm)

**ما هي الشجرة؟:**

تمكّنك هذه الأداة من عرض البيانات ذات التركيب الشجريّ: بمعنى أنّ كلّ عنصر يحتوى على مجموعة من العناصر الفرعيّة، كلّ عنصر منها يحتوى على مجموعة أخرى من العناصر الفرعيّة وهكذا..

ولن تكون هذه الأداةُ غريبةً عليك، فأنت تستخدمها كثيرا في متصفّح الويندوز Explorer، حيث تظهر على اليسار، بحيث تعرض محرّكات القرص الصلب Drives، وبجوار كلّ منها علامة "+"، فإذا ضغطت أيّا منها، ينسدل ليعرض المجلّدات التي يحتويها، فإذا ضغطت أيّا من هذه المجلّدات، عرض لك المجلّدات الفرعيّة التي يحتويها، وهكذا....



وكلّ عنصر في الشجرة يسمّى "قطب (فرع)" Node.. ويسمّى أوّل قطب في الشجرة "جذر الشجرة" Root، وتسمّى الأقطاب التي يحتويها "الأقطاب الفرعيّة" Child Nodes.

ويمكن أن تكون الفروع نصوصا، أو أيّ نوعٍ من الكائنات.. ولا مانع من أن يتماثل فرعان في الاسم.

ولا بدّ أن تتوقّع أنّ كلّ فرع في الشجرة هو مجموعة Collection وذلك لأنّه يحتوى على مجموعة من العناصر الأخرى.. وكلّ عنصر في هذه المجموعة هو كائن من النوع "فرع الشجرة" TreeNode، والذي قد يحتوي بدوره على فروع أخرى... وهكذا.

**ملحوظة:**

هذه الأداة تمثّل التركيب البرمجيّ المعروف باسم Tree Data Structure.. فإذا أردت أن تتعامل مع هذا التركيب من الكود، دون الحاجة لواجهة الاستخدام التي يتفاعل معها المستخدم، فيمكنك أن تعرّف متغيّرا من هذا النوع كالتالي:

**Dim T As New TreeView()**

وبهذا يمكنك التعامل مع المتغيّر T من خلال الكود، بنفس الطرائق التي سنشرحها في هذا الفصل، لتستخدمه في تخزين ومعالجة البيانات ذات التركيب الشجريّ، دون أن تحتاج لعرض أداة الشجرة للمستخدم.

**خصائص الشجرة:**

**مربعات اختيار CheckBoxes:**

اجعل هذه الخاصيّة True ليظهر مربّع اختيار أمام كلّ عنصر من عناصر الشجرة، بحيث يمكن للمستخدم أن يختار أكثر من عنصر من عناصر الشجرة في نفس الوقت.

**تحديد الصف كاملا FullRowSelect:**

اجعل هذه الخاصيّة True إذا أردت أن يتمّ تحديد الصفّ ـ الذي يوجد به العنصر كاملا ـ مع قدرة المستخدم على تحديد العنصر بالضغط على عنوانه، أو الضغط بجوار عنوانه.. بطريقة أوضح: لن يقتصر التحديد على النصّ الذي يمثّل عنوان العنصر، بل سيكون طول التحديد بعرض الشجرة (بصراحة: جربت قيمتي الخاصيّة ولم ألاحظ أيّ اختلاف!!!!).

**ارتفاع العنصر ItemHeight:**

يمكنك بهذه الخاصيّة أن تقرأ أو تغيّر ارتفاع عناصر الشجرة.

**تحرير اللافتة LabelEdit:**

إذا جعلت هذه الخاصيّة True، فسيمكن تغيير عنوان العنصر باستخدام الوسيلة BeginEdit.

**تعقّب فعّال HotTracking:**

لو جعلت هذه الخاصيّة True، فسيتألّق لون أيّ عنصر يمرّ فوقه مؤشّر الفأرة، تماما كما يحدث مع الوصلات الفائقة Hyperlinks في صفحات الإنترنت.

**الهامش Indent:**

استخدم هذه الخاصيّة لتحديد طول الهامش الأفقيّ بين كل مستوى جديد في الشجرة، والمستوى الأعلى منه.

**عرض الخطوط ShowLines:**

لو جعلت هذه الخاصيّة True، فسيتمّ توصيل كلّ عنصر فرعيّ والعنصر الذي يحتويه بخطّ.

**عرض علامتي "+" و"-" ShowPlusMinus:**

لو جعلت هذه الخاصيّة True، فسيتمّ عرض علامة "+" أمام العنصر الذي يحتوى على عناصر فرعيّة غير مسدلة، وعلامة "-" أمام العنصر الذي يحتوى على عناصر فرعيّة مسدلة (بصراحة لا أفهم هدف هذه الخاصيّة.. إنّ وجود هاتين العلامتين أمر حتميّ ليعلم المستخدم بوجود عناصر فرعيّة!).

**عرض خطوط الجذر ShowRootLines:**

لو جعلت هذه الخاصيّة True، فسيتم التوصيل بين عناصر الشجرة وجذرها بخطّ.

**مرتّبة Sorted:**

لو جعلت هذه الخاصيّة True، فسيتمّ ترتيب عناصر الشجرة تلقائيّا.. هذا الترتيب سيتمّ بطريقة مستقلّة لمجموعات العناصر التي تنتمي لكلّ فرع (وهل توجد طريقة أخرى؟؟).

**النصّ Text:**

تعيد لك النصّ الذي يمثّل عنوان العنصر المحدّد حاليّا في الشجرة.

**أعلى فرع TopNode:**

عندما يزداد عدد العناصر المعروضة في الشجرة عن طولها، فإنّها تعرض منزلقا رأسيّا.. في هذه الحالة يمكنك استخدام هذه الخاصيّة لتعرف الفرع الذي هو أعلى فرع ظاهر للمستخدم في الشجرة في هذه اللحظة.. طبعا تتوقّع أنّ هذه الخاصيّة تعيد كائنا من النوع TreeNode.

**عدد العناصر المرئيّة VisibleCount:**

تخبرك هذه الخاصيّة بعدد العناصر التي تظهر الآن للمستخدم في مساحة الشجرة.

**بعض وسائل الشجرة:**

**إسدال الكلّ ExpandAll:**

لإسدال كلّ عناصر الشجرة في كلّ مستوياتها مرّة واحدة.

**تقليص الكلّ CollapseAll:**

لتقليص كلّ عناصر الشجرة المسدلة في كلّ مستوياتها مرّة واحدة.

**معرفة العنصر من إحداثيّاته GetNodeAt:**

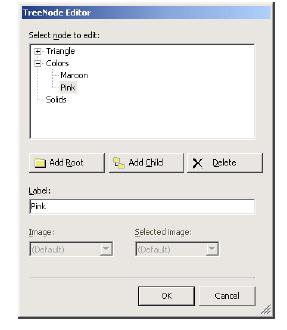
أرسل لهذه الوسيلة الإحداثيين الأفقيّ والرأسيّ، لتخبرك بالعنصر الموجود في الشجرة في هذا الموضع.. استفِد من هذا في حدث حركة الفأرة، لمعرفة العنصر الذي يقع حاليّا تحت مؤشّر الفأرة.

**معرفة عدد الفروع GetNodeCount:**

استخدم هذه الوسيلة لمعرفة عدد عناصر الشجرة.. ولهذه الوسيلة معامل واحد، إذا جعلته True، تعيد لك هذه الوسيلة عدد كلّ عناصر الشجرة مهما تشعّبت لأيّ مستوى.. وإذا جعلته False تعيد لك عدد عناصر المستوى الأوّل في الشجرة (الجذور فقط).

**إضافة العناصر للشجرة في وقت التصميم:**

كلّ ما ستفعله هو أن تحدّد خاصيّة Nodes في نافذة الخصائص، ثمّ اضغط زرّ الانتقال ليظهر لك نافذة محرّر عناصر الشجرة.



اضغط زرّ Add Root لإضافة عنصر رئيسيّ في الشجرة.. سيحمل أوّل عنصر الاسم الافتراضيّ Node0.. ولتغيير هذا الاسم، ستجد مربّع نصّ في أسفل النافذة معنونا بكلمة Label اكتب فيه النصّ الذي تريده له أن يظهر كعنوان للعنصر.. وهناك طريقة أخرى، هي أن تضغط العنصر نفسه ضغطة واحدة لتحديده، وبعد ثانيةٍ تضغطه ضغطة أخرى.. ستجد أنّ اسم العنصر قد تمّ تحديده وصار بإمكانك كتابته وتعديله.

بنفس هذه الطريقة يمكنك أن تضيف أيّ عدد من الجذور "العناصر الرئيسيّة".

ولكن ماذا عن العناصر الفرعيّة؟

لإضافة عنصر فرعيّ لأيّ عنصر في أيّ مستوى، حدّد هذا العنصر بضغطه بالفأرة واضغط زر Add Child.

وبهذه الطريقة يمكنك إضافة أيّ عدد من العناصر في أيّ عدد من المستويات.

ولحذف أيّ عنصر من أيّ مستوى، حدّد هذا العنصر بضغطه بالفأرة، ثمّ اضغط زر Delete.. ولكن حاذر، فعند حذف عنصر، يتمّ حذف كلّ ما تحته من عناصر، وهو ما لن يمكنك التراجع عنه.

**التعامل مع العناصر من الكود:**

للتعامل مع العناصر من الكود، استخدم مجموعة أقطاب الشجرة TreeNodeCollection كالتالي:

**TreeView1.Nodes**

هذه المجموعة تحتوى على عناصر الشجرة، وتمنحك العديد من الوسائل للتعامل معها.

ولكن ما نوع العناصر الموجودة بهذه المجموعة؟

إنّ كلّ عنصر في الشجرة هو كائن من النوع TreeNode، وهو كائن يحمل كافة المعلومات المطلوبة عن العنصر، ويمنحك الوسائل اللازمة للتعامل معه، كما يمنحك أيضا مجموعة أقطاب الشجرة TreeNodeCollection تحتوى على عناصره الفرعيّة.

دعنا نرى أولا كيف نتعامل مع مجموعة الفروع، ثمّ سنتفرّغ بعد ذلك للكائن TreeNode.

والسؤال الآن: كيف نعبّر عن أيّ عنصر في الشجرة؟

من خلال معرفتنا بالمجموعات Collections، فإنّنا نتوقّع أنّ التعبير التالي يمثّل أوّل عنصر في الشجرة:

**TreeView.Nodes.Item(0)**

والذي نعلم علم اليقين أنّه مكافئ للتعبير المختصر التالي:

**TreeView.Nodes(0)**

معنى هذا أنّك تستطيع قراءة عنوان العنصر بتعبير كالتالي:

**TreeView1.Nodes(0).Text**

ولكن ماذا لو أردت أن تتعامل مع العنصر الفرعيّ الأوّل، تحت العنصر الرئيسيّ الأوّل في الشجرة؟

في هذه الحالة استخدم جملة كالتالية:

**TreeView1.Nodes(0).Nodes(0).Text**

ولو أردت أن تتعامل مع عنصر في المستوى الثالث، فاستخدم تعبيرا كالتالي:

**TreeView1.Nodes(0).Nodes(0).Nodes(1).Text**

وهكذا.... .

وحتّى لا تبدو لك هذه التراكيب معقّدة، يمكنك تفتيتها إلى متغيّرات كالتالي:

**Dim Level1\_0 As TreeNode = TreeView1.Nodes(0)**

**Dim Level2\_0 As TreeNode = Level1\_0.Nodes(0)**

**Dim Level3\_0 As TreeNode = Level2\_0.Nodes(0)**

**MsgBox(Level3\_0.Text)**

ومن الجدير ذكرا أنّك تستطيع معرفة العنصر المحدّد حاليّا في الشجرة، باستخدام التعبير:

**TreeView1.CurrentNode**

ولمجموعة أقطاب الشجرة TreeNodeCollection الخصائص والوسائل التقليديّة للمجموعات Collection، وسنكتفي هنا بإيضاح وسلتي الإضافة والحذف:

**إضافة Add:**

هذه هي الوسيلة التي تضيف بها العناصر لفرع من أقطاب الشجرة.. وأبسط صيغة لها هي:

**newNode = Nodes.Add(عنوان العنصر)**

حيث newNode هو متغيّر من النوع TreeNode، يمكنك من خلاله التعامل مع العنصر الذي تمّت إضافته.

وهناك صيغة أخرى، تمكّنك من إضافة كائن من النوع فرع الشجرة TreeNode كالتالي:

**Dim nodeObj As New TreeNode()**

**nodeObj.Text = "عنصر جديد"**

**nodeObj.ForeColor = Color.BlueViolet**

**TreeView1.Nodes.Add(nodeObj)**

أمّا الصيغة الأخيرة، فهي تمكّنك من تحديد الموضع الذي سيضاف فيه العنصر الجديد:

**newNode = Nodes.Add(رقم العنصر, nodeObj)**

لا تنسَ أنّ العنصر الذي تضيفه سيندرج تحت العنصر الذي يمثّله مجموعة الفروع Nodes.. فمثلا، الجملة التالية تضيف عنصرا جديدا في المستوى الثاني:

**TreeView1.Nodes(0).Nodes.Add("عنصر جديد")**

**حذف Remove:**

يمكنك أن تحذف أيّ عنصر بما يندرج تحته من عناصر بجملة كالتالية:

**Nodes.Remove(رقم العنصر في مستواه)**

حيث المستوى يمثلّه المتغيّر Nodes.

**فرع الشجرة TreeNode**

عرفنا أنّ هذا هو الكائن الذي يمثّل نوع كلّ عنصر من عناصر الشجرة.. الآن نتعرّف عليه بطريقة أعمق:

* [**أهمّ خصائص الفرع:**](file:///C:\Users\Administrator\Desktop\vb.net\d-%20%20مواضيع%20متقدّمة\11-%20أدوات%20متقدّمة\7-%20فرع%20الشجرة\1-%20أهمّ%20خصائص%20الفرع.htm)
* [**خصائص التعديد Enumerating:**](file:///C:\Users\Administrator\Desktop\vb.net\d-%20%20مواضيع%20متقدّمة\11-%20أدوات%20متقدّمة\7-%20فرع%20الشجرة\2-%20خصائص%20التعديد.htm)
* [**خصائص مظهر العنصر:**](file:///C:\Users\Administrator\Desktop\vb.net\d-%20%20مواضيع%20متقدّمة\11-%20أدوات%20متقدّمة\7-%20فرع%20الشجرة\3-%20خصائص%20مظهر%20العنصر.htm)
* [**وسائل الفرع:**](file:///C:\Users\Administrator\Desktop\vb.net\d-%20%20مواضيع%20متقدّمة\11-%20أدوات%20متقدّمة\7-%20فرع%20الشجرة\4-%20وسائل%20الفرع.htm)

**أهمّ خصائص الفرع:**

**نصّ العنصر Text:**

يمكنك بهذه الخاصيّة أن تقرأ أو تغيّر عنوان العنصر.

**الحدود Bounds:**

تعيد لك هذه الخاصية مستطيلا Rectangle له أبعاد وموضع العنصر.

**تمّ اختياره Checked:**

يمكنك أن تقرأ أو تغيّر هذه الخاصيّة.. فإذا كانت قيمتها True، كان معنى ذلك أنّ أمام هذا العنصر علامة (✓).. طبعا لا بدّ أن تكون لخاصيّة CheckBoxesالخاصة بالشجرة القيمة True أولا، للسماح بعرض مربعات اختيار بجوار عناصر الشجرة.

**المسار الكامل FullPath:**

تعيد لك نصّا يعبّر عن المسار الكامل للعنصر الحالي، بدءا من الجذر الرئيسيّ ووصولا إلى العنصر.. في هذه النصّ ستوضع علامة فاصلة بين اسم كلّ عنصر وآخر.. هذه العلامة هي "\" في الوضع الافتراضيّ.. ولو شئت تغييرها، فاستخدم خاصيّة "فاصلة المسار" PathSeparator الخاصّة بأداة الشجرة نفسها.

**رقم Index:**

تعيد لك رقم العنصر في مجموعته.

**يتمّ تحريره IsEditing:**

ترجع هذه الخاصيّة True، إذا كان من الممكن تغيير اسم العنصر الحالي في هذه اللحظة.

**مسدل IsExpanded:**

عناصر الفرع الحالي مسدلة.

**إنّه محدّد IsSelected:**

ترجع هذه الخاصيّة True، إذا كان العنصر الحالي محدّدا Selected من قبل المستخدم.

**إنّه مرئيّ IsVisible:**

ترجع هذه الخاصيّة True، إذا كان العنصر الحالي ظاهرا أمام المستخدم دون أن يحتاج لتحريك المنزلق الرأسيّ لرؤيته.

**الفرع الرئيسيّ Parent:**

تعيد لك هذه الخاصيّة الفرع الرئيسيّ الذي ينتمي إليه هذا العنصر.

**الخاصيّة الإضافيّة Tag:**

هذه الخاصيّة من النوع Object، بحيث تمكّنك من تخزين أيّ معلومة (من أيّ نوع بيانات) ترتبط بالعنصر الحالي.

**خصائص التعديد Enumerating:**

للمرور عبر كلّ عناصر الفرع، يمكنك استخدام الخصائص التالية:

**أوّل فرع FirstNode:**

ترجع لك العنصر الأوّل في الفرع الحاليّ.

**الفرع التالي NextNode:**

ترجع لك العنصر التالي للعنصر الحالي في الفرع الحاليّ.

**الفرع المرئيّ التالي NextVisibleNode:**

ترجع لك العنصر التالي في الفرع الحاليّ، بشرط أن يكون ظاهرا أمام المستخدم دون الحاجة لتحريك المنزلق الرأسيّ.

**الفرع السابق PrevNode:**

ترجع لك العنصر السابق للعنصر الحالي في الفرع الحاليّ.

**الفرع المرئيّ السابق PrevVisibleNode:**

ترجع لك العنصر السابق في الفرع الحاليّ، بشرط أن يكون ظاهرا أمام المستخدم دون الحاجة لتحريك المنزلق الرأسيّ.

**الفرع الأخير LastNode:**

ترجع لك العنصر الأخير في الفرع الحاليّ.

والمثال التالي يريك كيف تطبع أسماء كلّ عناصر أحد الفروع، باستخدام الخصائص السابقة:

**Dim node As TreeNode**

**node = TreeView1.Nodes(0).Nodes(0).FirstNode**

**While node <> Nothing**

**Console.WriteLine(node.Text)**

**node = node.NextNode**

**End While**

**خصائص مظهر العنصر:**

تمنحك الشجرة القدرة على تغيير مظهر كلّ عنصر، عن طريق الخصائص التالية لفرع الشجرة TreeNode:

لون الخلفيّة **BackColor** ـ لون الخطّ **ForeColor** ـ خطّ الفرع **NodeFont**.

ليس هذا فحسب، بل إنّ في إمكانك وضع صورة لكلّ عنصر في الشجرة:

عندما تستخدم متصفّح الويندوز Explorer، فإنّك ترى أيقونة المجلّد أو محرّك الأقراص أمام كلّ عنصر في الشجرة.. إنّ بإمكانك أيضا أن تفعل ذلك مع شجرتك.

في البداية يجب أن تحمّل الصور اللازمة في الأداة "قائمة الصور" ImageList.

بعد هذا سنربط الشجرة بقائمة الصور، عن طريق وضع اسم قائمة الصور في خاصيّة ImageList الخاصّة بالشجرة.

بعد هذا سنستخدم الخاصيتين التاليتين من خصائص مجموعة الفروع:

**رقم الصورة ImageIndex:**

ضع في هذه الخاصيّة الرقم الذي يدلّ على موضع الصورة المطلوبة في قائمة الصور.. هذه الصورة سيتمّ عرضها أمام اسم العنصر.. الجدير بالذكر أنّ هذه الخاصيّة موجودة أيضا في الشجرة نفسها، وذلك حتّى تحدّد بها الصورة الافتراضيّة لكلّ عناصر الشجرة، بدلا من تحديدها لكل عنصر على حدة.

**رقم الصورة للعنصر المحدّد SelectedImageIndex:**

ضع في هذه الخاصيّة الرقم الذي يدلّ على موضع الصورة المطلوبة في قائمة الصور.. هذه الصورة سيتمّ عرضها أمام اسم العنصر عندما يكون محدّدا (عناصره الفرعيّة مسدلة).. فإذا لم تضع صورة في هذه الخاصيّة، فلن يتمّ تغيير صورة العنصر عند إسدال عناصره الفرعيّة.. هذه الخاصيّة موجودة أيضا في الشجرة نفسها، وذلك حتّى تحدّد بها صورة التحديد الافتراضيّة لكلّ عناصر الشجرة، بدلا من تحديدها لكل عنصر على حدة.

**وسائل الفرع:**

**بدء التحرير BeginEdit:**

تحدّد اسم العنصر حتّى يمكن للمستخدم إعادة تسميته.

**نهاية التحرير EndEdit:**

تنهي قدرة المستخدم على تغيير اسم العنصر.

**إسدال Expand:**

استخدم هذه الوسيلة لإسدال عناصر الفرع الحالي.

**إسدال الكلّ ExpandAll:**

استخدم هذه الوسيلة لإسدال عناصر الفرع الحالي، وما يتندرج تحتها من العناصر في أيّ مستوى.. وهذه الوسيلة موجودة أيضا للشجرة ككلّ، حيثُ تعمل على إسدال كلّ عناصر الشجرة مرّةً واحدة.

**تقليص Collapse:**

تقلّص عناصر الفرع الحالي لو كانت مسدلة.. ولا توجد وسيلة CollapseAll للفرع TreeNode، ولكنّها موجودة للشجرة نفسها، حيث تعمل على تقليص كلّ عناصر الشجرة دفعة واحدة.

**تحويل Toggle:**

تعكس وضع الفرع، فلو كانت عناصره مسدلة يتمّ تقليصها، ولو كانت غير مسدلة يتمّ إسدالها.

**تأكيد الرؤية EnsureVisible:**

تحرّك المنزلق الرأسي، بحيث يظهر الفرع أمام المستخدم لو كان غير ظاهر.

**عدد الفروع GetNodeCount:**

تخبرك بعدد العناصر المتفرّعة من هذا الفرع.

**إزالة Remove:**

تزيل الفرع بكلّ ما يندرج تحته من عناصر.. مثال:

**Dim Node As TreeNode**

**Node = TreeView1.Nodes(0).Nodes(7)**

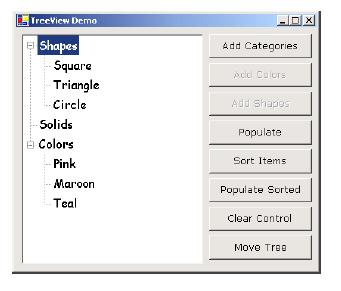
**Node.Remove**

ولحذف العنصر المحدّد حاليّا استخدم الجملة التالية:

**TreeView1.SelectedNode.Remove**

**مشروع TreeViewDemo:**

تعال ننشئ مشروعا بسيطا نستخدم فيه ما تعلّمناه عن الشجرة حتّى الآن، حتّى يتّضح لك أكثر.. والصورة التالية توضّح النموذج الرئيسيّ في التطبيق:



**زر "إضافة فئات" Add Categories:**

هذا الزرّ سيضيف عناصر أوّل مستوى في الشجرة.. وكوده كالتالي:

**TreeView1.Nodes.Add("Shapes")**

**TreeView1.Nodes.Add("Solids")**

**TreeView1.Nodes.Add("Colors")**

لاحظ أنّ المستخدم لو ضغط هذا الزرّ أكثر من مرّة، فستضاف هذه العناصر للشجرة أكثر من مرّة!!

**زر "إضافة ألوان" Add Colors:**

هذا الزرّ يضيف بعض الألوان تحت عنصر الألوان Colors:

**Dim cnode As TreeNode**

**cnode = TreeView1.Nodes(2)**

**cnode.Nodes.Add("Pink")**

**cnode.Nodes.Add("Maroon")**

**cnode.Nodes.Add("Teal")**

لاحظ أنّ هذه العناصر بعد إضافتها لن تنسدل تلقائيّا.. ولو شئت إسدالها فاستخدم الجملة التالية:

**cnode.Expand()**

لاحظ كذلك أنّ المستخدم لو ضغط هذا الزرّ قبل ضغط زرّ إنشاء الفئات، فسيحدث استثناء، لأنّ العناصر الرئيسيّة لم تنشأ بعد!

لهذا لا بدّ من جعل هذا الزرّ غير فعاّل حتّى تُنشأ العناصر الرئيسيّة أولا.

**زرّ تحريك الشجرة Move Tree:**

افترض الآن أنّك تريد إنشاء جذر رئيسي تتفرّع منه كلّ هذه الفئات.. في هذه الحالة اتبع الخطوات التالية:

* عرّف ثلاثة متغيّرات من النوع TreeNode وضع بها أقطاب الفئات الثلاثة.
* امسح كلّ محتويات الشجرة.
* أضف العنصر الرئيسي (وليكن اسمه Items).
* أنشئ تحت العنصر الرئيسيّ ثلاثة أقطاب وضع بها قيم المتغيّرات الثلاثة.

بهذا ستستعيد نفس الشجرة السابقة، ولكنّها ستكون متفرّعة من العنصر الرئيسيّ Items.

هذا هو الكود الذي يفعل ذلك:

**Dim colorNode, shapeNode, solidNode As TreeNode**

**' تخزين جذور الشجرة في المغيّرات**

**colorNode = TreeView1.Nodes(0)**

**shapeNode = TreeView1.Nodes(1)**

**solidNode = TreeView1.Nodes(2)**

**' محو محتويات الشجرة**

**TreeView1.Nodes.Clear()**

**' إضافة الجذر الجديد**

**TreeView1.Nodes.Add("Items")**

**' إضافة الجذور القديمة كعناصر تابعة للجذر الجديد**

**TreeView1.Nodes(0).Nodes.Add(colorNode)**

**TreeView1.Nodes(0).Nodes.Add(shapeNode)**

**TreeView1.Nodes(0).Nodes.Add(solidNode)**

ولو كان عدد الجذور كبيرا، فيمكنك أن تعدّل الكود السابق ليعرّف مصفوفة من الفروع تخزّن فيها كلّ الجذور، ثمّ تستخدم جملةً تكراريّة لملئها كالتالي:

**Dim TVList As New ArrayList()**

**Dim node As TreeNode**

**For Each node In TreeView1.Nodes**

**TVList.Add(node)**

**Next**

بعد ذلك يمكنك كتابة جملة تكراريّة لإضافة عناصر هذه المصفوفة القائمة للجذر الجديد.

**أهمّ أحداث الشجرة:**

تمنحك الشجرة العديد من الأحداث التي تتجاوب مع ما يقوم به المستخدم من أفعال.. وبجوار الأحداث المألوفة لكلّ أداة، لديك هذه الأحداث:

بعد اختيار العنصر **AfterCheck** ـ بعد تقليص عناصر الفرع **AfterCollapse** ـ بعد إسدال عناصر الفرع **AfterExpand** ـ بعد تحرير عنوان العنصر **AfterLabelEdit** ـ بعد تحديد العنصر **AfterSelect**.

وهناك مجموعة من الأحداث التي تحدث قبل إتمام المستخدم للحدث، ممّا يمنح لك الفرصة لإلغائه، حيث يمنحك المعامل e لهذه الأحداث خاصيّة "إلغاء" Cancel، التي لو جعلتها True يتمّ إلغاء الحدث:

**e.Cancel = True**

وهذه الأحداث هي:

قبل اختيار العنصر **BeforeCheck** ـ قبل تقليص عناصر الفرع **BeforeCollapse** ـ قبل إسدال عناصر الفرع **BeforeExpand** ـ قبل تحرير عنوان العنصر **BeforeLabelEdit** ـ قبل تحديد العنصر **BeforeSelect**.

لاحظّ أنّ هذه الأحداث تقع قبل حدوث الفعل.. إنّ هذا يعني أنّ الفرع الحالي CurrentNode ما زال كما هو لم يتغيّر، ويمكنك التعامل معه من داخل هذه الأحداث، حتّى لو كان الحدث الذي سيقع هو حدث تحديد عنصر آخر.

هذا ويمتلك المعامل e لكلّ الأحداث السابقة (التي تحدث قبل أو بعد) خاصيتين هامّتين، هما "الفرع" Node الذي وقع الحدث بسببه، ونوع الحدث Action الذي وقع.. ويفيدك المعامل الأخير عندما تكتب إجراءً واحدا يستجيب لأكثر من حدث مختلف (يجب أن تكون كلّها من نفس النوع "قبل" أو "بعد").. في هذه الحالة يمكنك أن تستخدم هذه الخاصيّة لمعرفة نوع الحدث الذي وقع.

ولديك في مجلّد برامج هذا الفصل المشروع Global.. افحصه جيّدا للتدرّب على التعامل مع الشجرة.

**المرور عبر كلّ عناصر الشجرة:**

افترض الآن أنّ لديك شجرةً لا تعرف مدى تشعّبها، وتريد كتابة كود عامّ يمرّ عبر كلّ عناصرها ليقوم بوظيفة ما (طباعة أسماء العناصر مثلا).. في هذه الحالة ليس أمامك إلا حلّ واحد فقط: استخدام إجراء ارتداديّ Recursive Procedure.. ولقد ذكرنا من قبل أنّه إجراءٌ عاديٌّ تماما، يماثلُ الإجراءات المعتادة في كلّ شيء، إلا إنّه يحتوى على سطر يستدعي نفسه ([راجع الفصل الرابع](file:///C:\Users\Administrator\Desktop\01-%20أساسيّات%20اللغة\04-%20مسار%20البرنامج%20وأجزاؤه\2-%20كتابة%20واستخدام%20الإجراءات\8-%20الإجراءات%20الارتدادية.htm)).

الآن سننشئ الإجراء ScanNode، والذي سنستدعيه دائما بإرسال أوّل جذر رئيسيّ كمعامل:

**ScanNode(TreeView1.Nodes(0))**

لو كانت الشجرة تحتوى على أكثر من جذر رئيسيّ، فاستخدم جملة تكراريّة كالتالية:

**Dim node As TreeNode**

**For Each node In TreeView1.Nodes**

**ScanNode(node)**

**Next**

ها هو ذا كود الإجراء ScanNode، وهو يقوم بإضافة عناصر الشجرة إلى قائمة ListBox مع إضافة بعض المسافات في بداية العنصر تدلّ على مستواه:

**Sub ScanNode(ByVal node As TreeNode)**

**Dim thisNode As TreeNode**

**Static IndentLevel As Integer**

**' اسمح للمستخدم بأداء العمليّات الأخرى، فقد يطول وقت هذا الإجراء**

**Application.DoEvents()**

**' \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**' اكتب هنا العمليّة التي تريد أداها.. وفي هذا المثال نضيف العناصر من الشجرة للقائمة**

**ListBox1.Items.Add(Space(IndentLevel) & node.Text)**

**' \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**' في الجزء التالي سيتمّ استدعاء نفس الإجراء لكلّ عنصر متفرّع من العنصر الحالي**

**' \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\***

**' شرط توقّف الإجراء الارتدادي هو ألا توجد عناصر متفرّعة من العنصر الحالي**

**' node.Nodes.Count = 0 أي:**

**If node.Nodes.Count > 0 Then**

**' سنستدعى عناصر مستوى جديد، لهذا سنزيد الهامش**

**IndentLevel += 5**

**For Each thisNode In node.Nodes**

**ScanNode(thisNode)**

**Next**

**' سينتهي هذا الإجراء ليعود لاستكمال المستوى السابق،**

**' لهذا لا بدّ من إنقاص الهامش**

**IndentLevel -= 5**

**End If**

**End Sub**

ويمكنك تجريب هذا في مشروع TreeViewScan في مجلّد برامج هذا الفصل.

ولو شئت صيغة عامّة للتعامل مع أيّ شجرة، فإليك هذا الطريقة:

ابدأ بتعريف مندوب Delegate عن الإجراء الذي سيقوم بالعمليّة التي تريد تنفيذها على عناصر الشجرة:

**Delegate Sub TreeAction (ByVal node As TreeNode)**

والآن عرّف الإجراء الذي سيمرّ عبر كلّ عناصر الشجرة، وأرسل له اسم الإجراء الذي سيقوم بالعمليّة المطلوبة كمعامل:

**Sub ScanNode(ByVal node As TreeNode, Action As TreeAction)**

**Dim thisNode As TreeNode**

**Call Action(node)**

**For Each thisNode In node.Nodes**

**ScanNode(thisNode)**

**Next**

**End Sub**

الآن يمكنك كتابة أيّ إجراء له معامل واحد من النوع TreeNode وإرساله كمعامل للإجراء ScanNode.. وكمثال، سنكتب الإجراء التالي لإضافة أسماء عناصر الشجرة إلى القائمة:

**Sub AddToList(ByVal node As TreeNode)**

**ListBox1.Items.Add(node.Text)**

**End Sub**

ولاستدعاء هذا الإجراء استخدم الكود التالي:

**Dim node As TreeNode**

**For Each node In TreeView1.Nodes**

**ScanNode(node, AddressOf AddToList)**

**Next**

ويمكنك كتابة أيّ عدد آخر من الإجراءات واستدعائها بطريقة مماثلة.

**ملحوظة:**

في الإجراء ScanNode يتمّ التعامل مع عناصر الشجرة بالترتيب (من الخارج للداخل.. من المستوى الأعلى للمستوى الأعمق).. أمّا لو شئت التعامل مع الشجرة بطريقة عكسيّة، بحيث يتمّ التعامل مع العناصر من الخارج (أطراف الشجرة) إلى الداخل (جذور الشجرة)، فكلّ ما عليك أن تفعله هو تغيير موضع استدعاء الإجراء Action بحيث يلي الجملة التكراريّة بدلا من أن يسبقها:

**Sub ScanNode(ByVal node As TreeNode, Action As TreeAction)**

**Dim thisNode As TreeNode**

**If node.Nodes.Count > 0 Then**

**For Each thisNode In node.Nodes**

**ScanNode(thisNode)**

**Next**

**End If**

**Call Action(node)**

**End Sub**

في هذه الحالة سيستمرّ الإجراء في استدعاء نفسه حتّى يصل إلى أوّل طرفٍ قصيّ في الشجرة.. عندئذٍ سيتحقّق شرط التوقّف (عدد العناصر الفرعيّة = 0)، وسيتمّ استدعاء الحدث Action لأوّل مرّة.. بعد هذا سيحين دور العنصر التالي لهذا العنصر في نفس الفرع (لو كان هناك واحد)، وسيستمرّ ذلك حتّى تنتهي عناصر هذا الفرع، حينئذ سينتقل التنفيذ لأوّل طرفٍ قصيّ في الفرع الذي يليه، وهكذا.. وتسمّى هذه الطريقة في المرور على عناصر الشجرة "العمق أولا" Depth First.. جرّب أن تغيّر موضع جملة إضافة العناصر للقائمة في مشروع TreeViewScan، وانظر كيف ستُضاف العناصر للقائمة.. (أفضل طريقة لفهم هذه الأمور هي تتبعها عمليّا أثناء تنفيذ VB لها).

**البحث عن أحد عناصر الشجرة:**

استكمالا لموضوع المرور عبر عناصر الشجرة، تعالوا نتعرّف على كيفيّة البحث عن عنصر في الشجرة (عن طريق نصّ العنصر) وتحديده..

للأسف: لا تمنحنا الشجرة وسيلة جاهزة لعمل ذلك.. ولن نستطيع استخدام خوارزميّة البحث السريع ها هنا، فللشجرة تركيب هرميّ، وليس خطّيا..

إذن فليس أمامنا إلا استخدام دالة ارتداديّة Recursive Function للبحث عن العنصر المطلوب في الشجرة.. فإن وجدته قامت بتحديده وإعادة مرجع له.. هذه هي الدالة التي تفعل هذا:

**Function SearchNode(ByVal CurrNode As TreeNode, \_**

**ByRef SearchElement As String) As TreeNode**

**' التأكّد من العنصر الحالي قبل البحث في عناصره الفرعيّة**

**If CurrNode.Text = SearchElement Then**

**T.SelectedNode = CurrNode**

**T.Focus()**

**Return CurrNode**

**End If**

**' البحث في العناصر الفرعيّة للعنصر الحاليّ**

**Dim node As TreeNode**

**For Each node In CurrNode.Nodes**

**Return SearchNode(node, SearchElement)**

**Next**

**End Function**

ولاستدعاء هذه الدالة، ضع شجرة على النموذج، وسمّها T.. وضع مربع نص (لتكتب به اسم العنصر الذي تريد البحث عنه في الشجرة) وزرا.. هذا هو كود زر البحث:

**Private Sub Button1\_Click(ByVal sender As System.Object, ByVal \_**

**e As System.EventArgs) Handles Button1.Click**

**Dim node , Result As TreeNode**

**For Each node In T.Nodes**

**Result = SearchNode(node, TextBox1.Text)**

**' إذا وجدنا العنصر، فلا داعي لإكمال البحث في باقي جذور الشجرة**

**If Not Result Is Nothing Then Exit For**

**Next**

**If Result Is Nothing Then MsgBox("عنصر البحث غير موجود")**

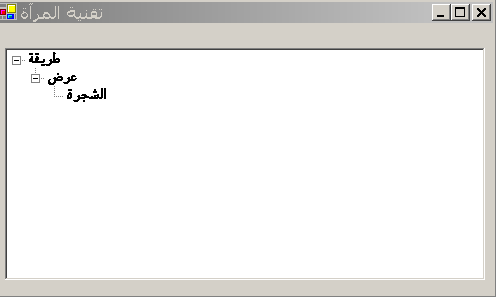
**End Sub**

**عرض الشجرة من اليمين لليسار[[1]](#footnote-1):**

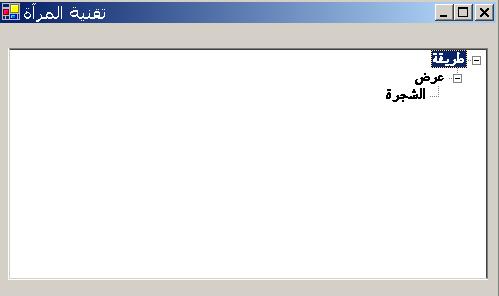
صحيح أنّ الشجرة تمتلك خاصيّة RightToLeft، ولكنّها لا تعمل كما نتوّقعها.. فهي تؤثّر فقط على طريقة عرض أسماء العناصر، وليس على طريقة عرض الشجرة ككلّ!

وللأسف: ظللنا ننتظر أن تحلّ ميكروسوفت هذه المشكلة عبر ثلاثة إصدارات متتالية.. فلم تمنحنا في النهاية إلا خاصيّة RightToLeft بذلك الأداء غير المُرضي!

هكذا تظهر الشجرة في الوضع العاديّ:



عموما.. لدينا حلّ، عبر استخدام مكتبات الربط الخاصة بالويندوز API.. هذا الحلّ يسمّى تقنية المرآة العاكسة Mirroring.. حيث سنقوم بعكس طريقة عرض الشجرة، لتبدو فروعها من اليمين لليسار.. هكذا:



**ملحوظة:**

العيب الرئيسيّ في هذه الطريقة، هو انعكاس التأثير المجسم.. حيث تلاحظ أنّ الخطّ الأسود الذي يوحي بعمق الأداة قد انتقل من اليسار لليمين.. طبعا هذا أخفّ الأضرار!

وأبسط حلّ لهذا، هو أن تستغني عن التجسيم تماما، وذلك بتغيير قيمة الخاصيّة "طراز الحافّة" BorderStyle إلى FixedSingle.

ولن أخوض في شرح دوال API، ولا طرق تعريفها.. فهذا موضوع مستقلّ بذاته..

وكلّ ما عليك هو أن تنقل جمل التعريف هذه في بداية النموذج (ستجد الكود في مشروع RightToLeftTree):

**Private Declare Function GetWindowLong Lib "user32" Alias \_**

**"GetWindowLongA" (ByVal hwnd As Integer, ByVal nIndex As \_**

**Integer) As Integer**

**Private Declare Function SetWindowLong Lib "user32" Alias \_**

**"SetWindowLongA" (ByVal hwnd As Integer, ByVal nIndex As \_**

**Integer, ByVal dwNewLong As Integer) As Integer**

**Const ws As Integer = &H400000**

**Const gw As Short = -20**

بعد ذلك انقل السطرين التاليين لحدث تحميل النموذج:

**Dim O As Integer = GetWindowLong(TreeView1.Handle.ToInt32, gw)**

**SetWindowLong(TreeView1.Handle.ToInt32, gw, O Or ws)**

خلاص.. صارت الشجرة تظهر من اليمين إلى اليسار!

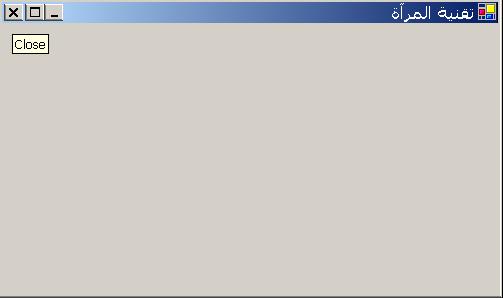
جدير بالذكر أنّ هذه الطريقة قابلة للاستخدام مع كلّ الأدوات.. وبهذا تستطيع حلّ مشاكلك مع مؤشّر التقدّم ProgressBar وشريط الأدوات ToolBar وأيّ أدوات خاصة أخرى لا تدعم العرض من اليمين لليسار.

وفي كلّ الأحول، كلّ ما عليك هو وضع اسم الأداة بدلا من TreeView1 في سطري الكود الأخيرين.. هذه هي الصيغة العامّة:

**Dim O As Integer = GetWindowLong(اسم الأداة.Handle.ToInt32, gw)**

**SetWindowLong(اسم الأداة.Handle.ToInt32, gw, O Or ws)**

بل والأطرف من هذا، أنّك تستطيع استخدام هذه التقنية مع النموذج نفسه.. في هذه الحالة سيتمّ عرضه كما يُعرض في الويندوز المعرّب، حيث ستظهر أزرار التكبير والتصغير والإغلاق على اليسار، وتظهر أيقونة التحكّم على اليمين.. كما سيؤدّي هذا أيضا لعكس وضع القوائم Menus!



جرّب الكود التالي في حدث تحميل النموذج:

**Dim O As Integer = GetWindowLong(Me.Handle.ToInt32, gw)**

**SetWindowLong(Me.Handle.ToInt32, gw, O Or ws)**

**حفظ عناصر الشجرة في ملفّ:**

تتوقّع أنّي سأقول لك إنّ الأمر في غاية البساطة، فكلّ ما عليك هو استخدام عمليّة السلسلة Serialization التي شرحناها في فصل "[التعامل مع المفّات والمجلّدات](file:///C:\Users\Administrator\Desktop\02-%20Classes%20الخلايا\10-%20الملفّات\2-%20القراءة%20من%20الملفّ%20والكتابة%20فيه\1-%20استخدام%20خلايا%20إطار%20العمل\8-%20خليّة%20التسلسل.htm)"، وبذلك يمكنك حفظ الشجرة وكلّ محتوياتها في ملفّ بجملة واحدة!

للأسف: ليس الأمر بهذه البساطة، فعملية السلسلة لا تستطيع حفظ الكائنات المتراكبة "الشجريّة"، التي يحتوى كلّ كائن منها على كائن آخر من نوعه، يحتوى على كائن ثالث من نوعهما .. وهكذا... وإنّما يمكن استخدام هذه العمليّة في حفظ الكائنات الخطّيّة، التي تحتوى على خصائص ووسائل، ولكن لا تمتلك تركيبا شجريّا متراكبا.

إذن ما العمل؟

يمكننا ببعض التحايل، أن نخزّن كلّ محتويات الشجرة في أيّ مجموعة قابلة للسلسلة (مثل المصفوفة القائمة ArrayList)، ثمّ بعد ذلك نحفظ هذه المجموعة في ملفّ!

لاحظ أنّنا سنحتاج في هذه الطريقة ـ ليس لتخزين قيمة كلّ عنصر فحسب ـ بل كذلك مستوى كلّ عنصر في الشجرة، وذلك حتّى نستطيع أن نعيد كلّ عنصر لفرعه في الشجرة عند تحميل العناصر من الملفّ مرّة أخرى.

من أجل هذا سنعرّف سجلا قابلا للسلسلة Serializable، بحيث يحتوى على خاصيّتين: نصّ العنصر ومستواه في الشجرة:

**<Serializable()> Structure sNode**

**Dim node As String**

**Dim level As Integer**

**End Structure**

لاحظ استخدامنا للسمة Serializable قبل تعريف السجلّ، وذلك حتّى يكون قابلا للسلسلة عند تخزينه في المصفوفة القائمة.

وبالنسبة لمستوى العنصر، فسيكون صفرا إذا كان العنصر أحد جذور الشجرة، و 1 إذا كان العنصر أحد العناصر المتفرّعة من جذر الشجرة، و 2 إذا كان العنصر متفرّعا من عنصر مستواه = 1، وهكذا... مستوى كلّ عنصر يزيد عن مستوى الفرع الذي يحتويه بواحد.

وعند تحميل الملفّ ستستخدم هذا المستوى، فكلّما زادت قيمته أضفنا عنصرا فرعيّا جديدا للعنصر الحاليّ، وكلّما نقص عدنا إلى الفرع الذي يحتوى على العنصر الحاليّ... وهكذا.

والآن سنكتب كود حفظ الشجرة.. إنّ كلّ ما سنفعله، هو كتابة إجراء اسمه SaveTree لإضافة نصّ العنصر المرسل إليه ومستواه في سجل من النوع sNode، وإضافة هذا السجلّ إلى المصفوفة القائمة TreeList، التي سنعرّفها على مستوى النموذج كالتالي:

**Dim TreeList As New ArrayList()**

ها هو ذا كود الإجراء SaveTree:

**Sub SaveTree(ByVal node As TreeNode)**

**Dim myNode As sNode**

**myNode.level = node.FullPath.Split("\"c).GetUpperBound(0)**

**myNode.node = node.Text**

**TreeList.Add(myNode)**

**End Sub**

هل أدركت ما يفعله السطر الثاني في الإجراء السابق؟

إنّه يستخدم خاصيّة المسار الكامل للعنصر FullPath ليستنتج منها مستواه، وذلك بتقطيع هذا المسار إلى عناصره.. هذه العناصر تفصل بينها في المسار العلامة "\".. إنّ العناصر الناتجة ستوضع في المصفوفة التي تعيدها الدالة Split.. رقم الخانة الأخيرة في هذه المصفوفة هو مستوى العنصر (لأنّه أقلّ من عدد الخانات بواحد).

الآن كلّ ما علينا فعله هو استدعاء الإجراء ScanNode ـ الذي كتبناه من قبل ـ وإرسال الإجراء SaveTree كمعامل له، بحيث توضع كلّ عناصر الشجرة في المصفوفة القائمة:

**Dim RootNode As TreeNode**

**For Each RootNode In TreeView1.Nodes**

**ScanNode(RootNode, AddressOf SaveTree)**

**Next**

والآن لم يتبقّ إلا حفظ المصفوفة القائمة بالطريقة التي تعلّمناها من قبل.. افترض أنّ FileName هو متغيّر يحتوى على اسم ومسار الملفّ الذي اختاره المستخدم بواسطة مربّع حوار "حفظ ملفّ".. هذا هو الكود الذي يحفظ المصفوفة القائمة:

**Dim formatter As New BinaryFormatter**

**Dim saveFile As FileStream**

**saveFile = File.Create(FileName)**

**formatter.Serialize(saveFile, TreeList)**

**saveFile.Close()**

ولتحميل هذه البيانات من الملفّ إلى المصفوفة القائمة مرّة أخرى، استخدم الكود التالي:

**Dim formatter As New BinaryFormatter**

**Dim openFile As FileStream**

**openFile = File.Open(FileName, FileMode.Open)**

**TreeList = CType(formatter.Deserialize(openFile), ArrayList)**

**openFile.Close()**

لم يبقَ إلا إعادة العناصر إلى أفرعها على الشجرة.. هذا هو الكود الذي يقوم بهذا:

**Dim o As sNode**

**Dim currNode As TreeNode**

**Dim CurrLevel As Integer = 0**

**Dim i As Integer**

**For i = 0 To TreeList.Count - 1**

**o = CType(TreeList(i), sNode)**

**If o.level = CurrLevel Then**

**If CurrLevel = 0 Then**

**' هذا العنصر من جذور الشجرة**

**currNode = TreeView1.Nodes.Add(o.node)**

**Else ' العنصران في مستوى واحد،**

**' أضف العنصر الجديد لنفس فرع العنصر الحالي**

**' العنصر الذي سنضيفه سيصبح هو العنصر الحالي**

**currNode = currNode.Parent.Nodes.Add(o.node)**

**End If**

**ElseIf o.level > CurrLevel Then**

**' مستوى العنصر الجديد أكبر من مستوى العنصر الحالي**

**' إذن فالعنصر الجديد متفرّع من العنصر الحالي**

**' وسيصبح العنصر الجديد بعد إضافته هو العنصر الحالي**

**currNode = currNode.Nodes.Add(o.node)**

**' لا تنس أن تزيد مستوى العنصر الحالي بواحد**

**CurrLevel = o.level**

**Else ' إذا كان مستوى العنصر الجديد أصغر من المستوى الحالي**

**' فيجب أن ننقص المستوى الحالي بمقدار الفرق بينَهما**

**' إلى أن نصل للفرع الذي سنضع فيه العنصر الجديد**

**While o.level <= CurrLevel**

**currNode = currNode.Parent**

**CurrLevel -= 1**

**End While**

**' أضف العنصر الجديد للفرع الذي وصلنا إليه،**

**' واجعل العنصر الجديد هو العنصر الحالي**

**currNode = currNode.Nodes.Add(o.node.ToString)**

**' لا تنس أن تزيد مستوى العنصر الحالي بواحد**

**CurrLevel = o.level**

**End If**

**Next**

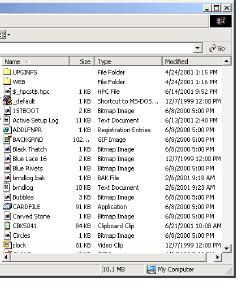
ويحتوى النموذج الرئيسيّ في المشروع Globale في مجلّد برامج هذا الفصل على القائمة File، التي تمنحك الأمران Save و Load، وهما يقومان بالوظيفتين اللتين شرحناهما توّا.. الجدير بالذكر أنّ هذا التطبيق قد وضع دالتي الحفظ والتحميل في خليّة Class، حتّى يمكنك استخدام هذه الخليّة بسهولة من أيّ مشروع آخر.

**الأداة "قائمة العرض" ListView Control**

* [**ما هي قائمة العرض؟:**](file:///C:\Users\Administrator\Desktop\vb.net\d-%20%20مواضيع%20متقدّمة\11-%20أدوات%20متقدّمة\4-%20قائمة%20العرض\0-%20ما%20هي%20قائمة%20العرض.htm)
* [**أهمّ خصائص قائمة العرض:**](file:///C:\Users\Administrator\Desktop\vb.net\d-%20%20مواضيع%20متقدّمة\11-%20أدوات%20متقدّمة\4-%20قائمة%20العرض\1-%20أهمّ%20خصائص%20قائمة%20العرض.htm)
* [**أهمّ وسائل قائمة العرض:**](file:///C:\Users\Administrator\Desktop\vb.net\d-%20%20مواضيع%20متقدّمة\11-%20أدوات%20متقدّمة\4-%20قائمة%20العرض\2-%20أهمّ%20وسائل%20قائمة%20العرض.htm)
* [**أهمّ أحداث قائمة العرض:**](file:///C:\Users\Administrator\Desktop\vb.net\d-%20%20مواضيع%20متقدّمة\11-%20أدوات%20متقدّمة\4-%20قائمة%20العرض\3-%20أهمّ%20أحداث%20قائمة%20العرض.htm)
* [**إضافة العناصر لقائمة العرض في وقت التصميم:**](file:///C:\Users\Administrator\Desktop\vb.net\d-%20%20مواضيع%20متقدّمة\11-%20أدوات%20متقدّمة\4-%20قائمة%20العرض\4-%20إضافة%20العناصر%20في%20وقت%20التصميم.htm)
* [**التعامل مع عناصر القائمة من الكود:**](file:///C:\Users\Administrator\Desktop\vb.net\d-%20%20مواضيع%20متقدّمة\11-%20أدوات%20متقدّمة\4-%20قائمة%20العرض\5-%20التعامل%20مع%20عناصر%20القائمة%20من%20الكود.htm)
* [**ترتيب عناصر قائمة العرض:**](file:///C:\Users\Administrator\Desktop\vb.net\d-%20%20مواضيع%20متقدّمة\11-%20أدوات%20متقدّمة\4-%20قائمة%20العرض\11-%20ترتيب%20عناصر%20قائمة%20العرض.htm)
* [**التعامل مع العناصر المحدّدة في القائمة:**](file:///C:\Users\Administrator\Desktop\vb.net\d-%20%20مواضيع%20متقدّمة\11-%20أدوات%20متقدّمة\4-%20قائمة%20العرض\12-%20التعامل%20مع%20العناصر%20المحدّدة%20في%20القائمة.htm)
* [**تدريب:**](file:///C:\Users\Administrator\Desktop\vb.net\d-%20%20مواضيع%20متقدّمة\11-%20أدوات%20متقدّمة\4-%20قائمة%20العرض\13-%20تدريب.htm)
* [**حفظ محتويات قائمة العرض في ملفّ:**](file:///C:\Users\Administrator\Desktop\vb.net\d-%20%20مواضيع%20متقدّمة\11-%20أدوات%20متقدّمة\4-%20قائمة%20العرض\14-%20حفظ%20محتويات%20قائمة%20العرض%20في%20ملفّ.htm)

**ما هي قائمة العرض؟:**

تماثل هذه الأداة القائمة العاديّة ListBox، إلا إنّها تمتلك القدرة على عرض عناصرها بأكثر من طريقة مختلفة، مع قدرتها على عرض العناصر المتفرّعة عن كلّ عنصر (في صور أعمدة وليس في صورة شجرة).. وهي أيضا ليست غريبةً عليك، فهي التي تعرض لك المجلّدات والملفّات في متصفّح الويندوز Explorer.



**أهمّ خصائص قائمة العرض:**

هذه هي الخصائص التي تتحكّم في مظهر قائمة العرض:

**عرض View:**

استخدم هذه الخاصيّة لتختار كيف سيتمّ عرض العناصر في القائمة.. ولهذه الخاصيّة القيم التالية:

|  |  |
| --- | --- |
| LargeIcon | أيقونات كبيرة، تحت كلّ منها اسم العنصر.. وهذه هي القيمة الافتراضيّة. |
| SmallIcon | أيقونات صغيرة، بجوار كلّ منها اسم العنصر. |
| List | قائمة بأسماء العناصر، مرتصّة عموديّا من أعلى لأسفل. |
| Report | يتمّ عرض اسم العنصر في عمود، مع وجود عدّة أعمدة أخرى تمثّل بيانات العنصر. |

**تنظيم العناصر Arrange:**

استخدم هذه الخاصيّة لتختار كيفيّة ترتيب العناصر في قائمة العرض.. وهي تأخذ القيم التالية:

|  |  |
| --- | --- |
| افتراضيّ Default | عندما يتمّ تحريك العنصر (بسحبه بالفأرة) فإنّه يبقى في مكانه الجديد. |
| يسار Left | يتمّ محاذاة العناصر إلى جهة اليسار. |
| محاذاة الشبكة SnapToGrid | عندما يحرّك المستخدم العنصر، فإنّه يستقرّ في أقرب نقطةٍ لخطوط شبكة غير مرئيّة على الأداة. |
| أعلى Top | يتمّ محاذاة العناصر إلى الجهة العليا من الأداة. |

**تنظيم تلقائيّ AutoArrange:**

إذا جعلت هذه الخاصيّة True، فسيتمّ إعادة ترتيب العناصر تلقائيّا لو قام المستخدم بحذف أو تحريك أحدها.

**طراز عناوين الأعمدة HeaderStyle:**

عند عرض العناصر بتفاصيلها (خاصيّة View بالقيمة Report)، فإنّ هذه الخاصيّة تستخدم لتحديد طراز عناوين الأعمدة التي يتمّ فيها عرض العناصر وتفاصيلها.. وقيم هذه الخاصيّة هي:

|  |  |
| --- | --- |
| Clickable | يتم عرض عناوين الأعمدة مع قدرة المستخدم على ضغطها بالفأرة، ليتمّ ترتيب عناصرها تبعا لمحتويات العمود المضغوط. |
| Nonclickable | يتم عرض عناوين الأعمدة لكنّ المستخدم لا يستطيع ضغطها بالفأرة. |
| None | لا يتمّ عرض عناوين الأعمدة. |

**السماح بترتيب الأعمدة AllowColumnReorder:**

إذا جعلت هذه الخاصيّة True، فسيكون باستطاعة المستخدم أن يسحب العمود (بضغط عنوانه) ليحرّكه إلى أيّ موضع، بحيث يغيّر ترتيبه بينَ الأعمدة.

**التنشيط Activation:**

استخدم هذه الخاصيّة لتحدّد الفعل الذي ينشّط عناصر القائمة، من بين الأفعال التالية:

|  |  |
| --- | --- |
| OneClick | يتمّ تنشيط العنصر بضغطة واحدة. |
| Standard | يتمّ تنشيط العنصر بنقره مرّتين بالفأرة. |
| TwoClick | يتمّ تنشيط العنصر مع تغيّر لون نصّه بنقره مرّتين بالفأرة. |

**تحديد الصفّ بالكامل FullRowSelect:**

إذا جعلت هذه الخاصيّة True، فسيتمكّن المستخدم من تحديد العنصر بكلّ تفاصيله (أعمدته) عند ضغط أيّ خانة في أيّ عمود.

**خطوط الشبكة GridLines:**

إذا جعلت هذه الخاصيّة True، فسيتمّ عرض خطوط فاصلة بين العناصر وتفاصيلها.

**تحرير اللافتة LabelEdit:**

إذا جعلت هذه الخاصيّة True، فسيتمكّن المستخدم من تغيير اسم العنصر.

**التفاف اللافتة LabelWrap:**

إذا جعلت هذه الخاصيّة True، فسيتمّ التفاف العناوين الطويلة للعناصر إلى السطر التالي، وذلك أثناء عرض العناصر كأيقونات.

**تحديد بالإشارة HoverSelection:**

إذا جعلت هذه الخاصيّة True، فسيتمكّن المستخدم من تحديد أيّ عنصر بمجرّد الإشارة إليه بالفأرة دون ضغطه.

**متعدّدة التحديد MultiSelect:**

إذا جعلت هذه الخاصيّة True، فسيتمكّن المستخدم من تحديد أكثر من عنصر في نفس الوقت، بنفس الطرق التي يحدّد بها العناصر في متصفّح الويندوز Explorer (أعتقد انّك تعرفها).

**مربّعات اختيار CheckBoxes:**

إذا جعلت هذه الخاصيّة True، فسيتمّ عرض مربّع اختيار بجوار كلّ عنصر.. وللتعامل مع العناصر المختارة Selected، يمكنك استخدام الخاصيتين: "أرقام العناصر المختارة" CheckedIndices و"العناصر المختارة" CheckedItems.. الطريف أنّ بإمكانك اختيار أيّ شكل تريده بدلا من مربّع الاختيار التقليديّ، ولفعل ذلك، استخدم خاصيّة "قائمة صور الحالة" StateImageList، حيث يمكنك أن تضع بها اسم نسخة من الأداة "قائمة الصور" ImageList.. قائمة الصور هذه يجب أن تحتوى على صورتين على الأقلّ: الصورة الأولى (رقم 0) سيتمّ عرضها بديلا لمربّع الاختيار في وضع عدم الاختيارUnchecked ، والصورة الثانية (رقم 1) سيتمّ عرضها بديلا لمربّع الاختيار في وضع الاختيار Checked.. وفي كلّ الأحوال لن يؤثّر ذلك على أيقونة العنصر.

**العنصر الفعّال FocusedItem:**

استخدم هذه الخاصيّة لتعرِف العنصر الفعال حاليا في القائمة، ذلك الذي يستقبل ضربات أزرار لوحة المفاتيح.

**أعلى عنصر TopItem:**

استخدم هذه الخاصيّة لمعرفة أعلى عنصر مرئيّ في القائمة.

**ترتيب Sorting:**

تحدّد كيفيّة ترتيب عناصر القائمة، وهي تأخذ القيم التالية:

|  |  |
| --- | --- |
| None | بدون ترتيب. |
| Ascending | ترتيب العناصر تصاعديّا. |
| Descending | ترتيب العناصر تنازليّا. |

ونظرا لأنّ القائمة قد تحتوي على أكثر من عمود، يمكن أن ترتّبها على أساس أيّ منها، فإنّ ترتيب القائمة يحتاج لنقاشٍ أطول، سنفرد له [مقطعا مستقلا](file:///C:\Users\Administrator\Desktop\vb.net\d-%20%20مواضيع%20متقدّمة\11-%20أدوات%20متقدّمة\11-%20ترتيب%20عناصر%20قائمة%20العرض.htm) لاحقا إن شاء الله.

**أهمّ وسائل قائمة العرض:**

**ترتيب الأيقونات ArrangeIcons:**

عند عرض العناصر كأيقونات، استخدم هذه الوسيلة لإعادة ترتيبها.

**تأكّد من أنّه مرئيّ EnsureVisible:**

استخدم هذه الوسيلة لتتأكّد من أنّ المستخدم يرى العنصر الذي ترسل رقمه كمعامل، دون أن يضطرّ إلى تحريك المنزلق الرأسيّ أو الأفقيّ.

**معرفة العنصر من إحداثيّاته GetItemAt:**

أرسل لهذه الوسيلة الإحداثيّات التي تريد أن تعرف العنصر الموجود فيها.. ويمكنك الاستفادة من هذه الخاصيّة في حدث حركة الفأرة، لمعرفة العنصر الذي تمرّ فوقه الفأرة في هذه اللحظة.

**معرفة إحداثيات العنصر GetItemRect:**

تعيد لك هذه الوسيلة المستطيل الذي يحدّد موضع العنصر وإحداثيّاته.

**ترتيب Sort:**

لترتيب عناصر القائمة.

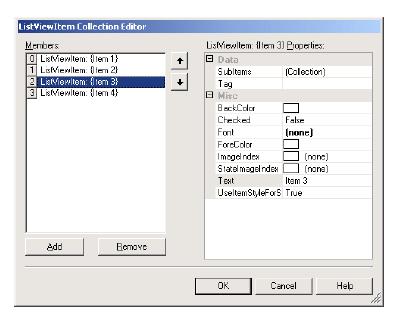
**أهمّ أحداث قائمة العرض:**

لتجدنّ هذه الأحداث واضحةً من أسمائها:

قبل تحرير اللافتة **BeforeLabelEdit** ـ بعد تحرير اللافتة **AfterLabelEdit** ـ ضغط القائمة **Click** ـ ضغط عنوان عمود **ColumnClick** ـ تنشيط عنصر **ItemActivate** ـ اختيار عنصر **ItemCheck** ـ سحب عنصر **ItemDrag** ـ تغيّر رقم العنصر المحدّد **SelectedIndexChanged**.

**إضافة العناصر لقائمة العرض في وقت التصميم:**

اضغط زر الانتقال المجاور لخاصيّة Items في نافذة الخصائص لتظهر لك نافذة محرّر عناصر القائمة.



هذه النافذة تنقسم إلى جزئين: على اليسار قائمة العناصر Items، حيث يمكنك أن تضغط زرّ الإضافة Add لإضافة عنصر جديد، أو تحدّد أيّ عنصر وتضغط زرّ الحذف Remove لإزالة العنصر المحدّد.. أمّا على اليمين، فتظهر قائمة بخصائص العنصر المحدّد حاليّا في قائمة العناصر.. وأهمّ هذه الخصائص هي خاصيّة Text التي تسمح لك بكتابة نصّ العنصر.. هذا بالإضافة إلى الخصائص التي تمكّنك من تغيير لون العنصر وخطّه وصورته.. إلخ، وسنتعرّف عليها حينما نشرح خصائص عنصر القائمة ListViewItem.

كما ستجد أنّ أوّل خاصيّة في هذه القائمة هي خاصيّة SubItems، وهي تمكّنك من إضافة العناصر الفرعيّة، التي ستظهر كتفاصيل للعنصر الحالي عند عرض الأعمدة في قائمة العرض.. ولإضافة عناصر فرعيّة (تفاصيل)، اضغط زر الانتقال المجاور لهذه الخاصيّة.. ستظهر لك نافذة مماثلة تماما للنافذة الحاليّة.. هذه النافذة تسمح لك بإضافة الأعمدة (العناصر الفرعية ـ التفاصيل) للعنصر، مع قدرتك على اختيار خطّ العمود ولونه.. أضف ما تريد من عناصر فرعيّة، ثمّ أغلق هذه النافذة للعودة إلى نافذة العناصر.

**التعامل مع عناصر القائمة من الكود**

لكي تتعامل مع عناصر القائمة، يجب أن تفهم هذا التركيب:

- تحتوى القائمة على مجموعة العناصر Items Collection وعلى مجموعة الأعمدة Columns Collection.

- تحتوى مجموعة العناصر Items Collection على عناصر من النوع "عنصر القائمة" ListViewItem.

- يحتوى عنصر القائمة على مجموعة العناصر الفرعيّة SubItems collection.

- تحتوى مجموعة العناصر الفرعيّة SubItems collection على عناصر من النوع ListViewSubItem.

وفيما يلي شرح كلّ واحد من هذه الكائنات:

* [**مجموعة العناصر Items Collection:**](file:///C:\Users\Administrator\Desktop\vb.net\d-%20%20مواضيع%20متقدّمة\11-%20أدوات%20متقدّمة\5-%20التعامل%20مع%20عناصر%20القائمة%20من%20الكود\6-%20مجموعة%20العناصر.htm)
* [**الكائن "عنصر القائمة" ListViewItem Object:**](file:///C:\Users\Administrator\Desktop\vb.net\d-%20%20مواضيع%20متقدّمة\11-%20أدوات%20متقدّمة\5-%20التعامل%20مع%20عناصر%20القائمة%20من%20الكود\7-%20عنصر%20القائمة.htm)
* [**مجموعة الأعمدة Columns Collection:**](file:///C:\Users\Administrator\Desktop\vb.net\d-%20%20مواضيع%20متقدّمة\11-%20أدوات%20متقدّمة\5-%20التعامل%20مع%20عناصر%20القائمة%20من%20الكود\8-%20مجموعة%20الأعمدة.htm)
* [**مجموعة العناصر الفرعيّة SubItems collection:**](file:///C:\Users\Administrator\Desktop\vb.net\d-%20%20مواضيع%20متقدّمة\11-%20أدوات%20متقدّمة\5-%20التعامل%20مع%20عناصر%20القائمة%20من%20الكود\9-%20مجموعة%20العناصر%20الفرعيّة.htm)
* [**كائن "العنصر الفرعيّ" ListViewSubItem:**](file:///C:\Users\Administrator\Desktop\vb.net\d-%20%20مواضيع%20متقدّمة\11-%20أدوات%20متقدّمة\5-%20التعامل%20مع%20عناصر%20القائمة%20من%20الكود\10-%20كائن%20العنصر%20الفرعيّ.htm)

**مجموعة العناصر Items Collection:**

تمتلك قائمة العرض الخاصيّة Items، التي تمكّنك من التعامل مع جميع عناصرها.. هذه الخاصيّة هي مجموعة Collection، ولا بدّ أنّك قد صرت خبيرا في التعامل مع المجموعات.. ولإضافة عنصر لهذه المجموعة، استخدم الوسيلة الشهيرة إضافة Add، بالصيغة التالية:

**ListView1.Items.Add(عنوان العنصر)**

وهناك صيغة أخرى تمكّنك من اختيار الصورة التي ستُعرض لهذا العنصر، وذلك بتحديد رقم الصورة في الأداة قائمة الصور ImageList:

**ListView1.Items.Add(عنوان العنصر, رقم الصورة)**

جديرٌ بالذكر أنّ عليك ربط قائمة العرض بقائمة الصور أولا عن طريق خاصيّة LargeImageList، وهي تحدّد الصورة التي ستظهر عند عرض العناصر كأيقونات كبيرة، وخاصيّة SmallImageList وهي تحدّد الصورة التي ستظهر عند عرض العناصر كأيقونات صغيرة.

كما توجد صيغة ثالثة، تمكّنك من إضافة العنصر من متغيّر من النوع ListViewItem.. انظر للمثال التالي:

**Dim LItem As New ListViewItem()**

**LItem.Text = "عنصر 1"**

**LItem.SetSubItem(0, "عنصر فرعيّ 1")**

**LItem.SetSubItem(1, "عنصر فرعيّ 2"**

**LItem.SetSubItem(2, "عنصر فرعيّ 3")**

**ListView1.ListItems.Add(LItem)**

وللتعامل مع عنصر من العناصر، يمكنك استخدام خاصيّة Item، كالتالي:

**MsgBox(ListView1.ListItems.Item(0).Text)**

أو بصورة مختصرة:

**MsgBox(ListView1.ListItems(0).Text)**

هذا وتمتلك مجموعة العناصر وسائلَ أخرى ستفهمها تماما من أسمائها، فهي مشابهة لوسائل عناصر القائمة العاديّة.. هذه الوسائل هي:

إدراج عنصر **Insert** ـ إضافة العناصر من مصفوفة **AddRange** ـ نسخ العناصر لمصفوفة **CopyTo** ـ إفراغ **Clear** ـ تحتوى على عنصر معيّن **Contains** ـ رقم عنصر معيّن **IndexOf** ـ حذف عنصر يمثّله متغيّر **Remove** ـ حذف عنصر برقمه **RemoveAt**.

**الكائن "عنصر القائمة" The ListViewItem Object:**

كلّ عنصر من عناصر القائمة هو من النوع ListViewItem.. وهذا الكائن يمتلك الخصائص التالية:

لون خلفيّة العنصر **BackColor** ـ لون خطّ العنصر **ForeColorـ** خطّ العنصر **Font** ـ نصّ العنصر **Text** ـ تمّ اختيار العنصر **Checked** ـ تمّ تحديد العنصر **Selected**.

**مجموعة العناصر الفرعيّة SubItems collection:**

يمتلك العنصر ListViewItem أيضا خاصيّة SubItems.. هذه الخاصيّة تمكّنك من التعامل مع بيانات أعمدة التفاصيل التابعة لهذا العنصر، سواء بإضافتها وحذفها، أو قراءتها وتغييرها.. فمثلا لقراءة العمود الثاني للصفّ الأوّل في القائمة، يمكنك استخدام الجملة التالية:

**Txt = ListView1.Items(0).SubItems(1).Text**

لاحظ أنّ العمود الأوّل يمكن التعامل معه بطريقتين:

Txt = **ListView1.Items(1).Text**

أو:

**Txt = ListView1.Items(1).SubItems(0).Text**

وبهذا يعتبر العنصر الفرعيّ رقم صفر هو العنصر نفسه، وذلك حتّى لا ترتبك في ترتيب الأعمدة!

ولإضافة عنصر فرعيّ استخدم الوسيلة الشهيرة لكلّ المجموعات Add، كالتالي:

**LItem.SubItems.Add("عنوان العنصر الفرعيّ")**

وهناك صيغة أخرى لهذه الوسيلة، تقبل كمعامل متغيّرا من النوع ListViewSubItem.

ولو أردت إدراج عنصر فرعيّ في موضع معيّن، فاستخدم وسيلة الإدراج Insert كالتالي:

**LItem.SubItems.Insert(موضع العنصر الفرعيّ, عنوان العنصر الفرعيّ)**

ما يجب أن تنتبه له جيّدا، هو أنّك يجب أن تنشئ أعمدة قائمة العرض أولا قبل أن تحاول إضافة العناصر الفرعيّة لكلّ عنصر، وذلك باستخدام وسيلة الإضافة Add الخاصّة بمجموعة عناوين الأعمدة Columns التابعة لخصائص قائمة العرض.. والمثال التالي يريك كيف تضيف عمودين للقائمة:

**ListView1.Columns.Add("الشركة", 20)**

**ListView1.Columns.Add("رقم الهاتف", 20)**

بعد هذا يمكنك أن تضيف أسماء الشركات كعناصر، وأرقام هواتفها كعناصر فرعيّة، كالتالي:

**Dim X As New ListViewItem()**

**X = ListView1.Items.Add("شركة 1")**

**X.SubItems.Add("3684643")**

ولإزالة عنصر فرعيّ استخدم الوسيلة RemoveAt، مرسلا لها رقم العنصر الفرعيّ كمعامل.. أمّا لو كان لديك متغيّر يشير إلى عنصر فرعيّ تريد حذفه، فيمكنك إرسال هذا المتغيّر إلى الوسيلة Remove لحذف هذا العنصر الفرعيّ.

هذا وتمتلك مجموعة العناصر الفرعيّة وسائلا أخرى ستفهمها تماما من أسمائها، فهي مشابهة لوسائل عناصر القائمة العاديّة.. هذا الوسائل هي:

إضافة العناصر الفرعيّة من مصفوفة **AddRange** ـ إفراغ **Clear** ـ تحتوى على عنصر فرعيّ معيّن **Contains** ـ رقم عنصر فرعيّ معيّن **IndexOf**.

**كائن "العنصر الفرعيّ" ListViewSubItem:**

هذا وكلّ عنصر فرعيٍّ هو من النوع ListViewSubItem، وهو يمتلك الخصائص التالية:

لون خلفيّة العنصر الفرعيّ **BackColor** ـ لون خطّ العنصر الفرعيّ **ForeColor** ـ خطّ العنصر الفرعيّ **Font** ـ نصّ العنصر الفرعيّ **Text**.

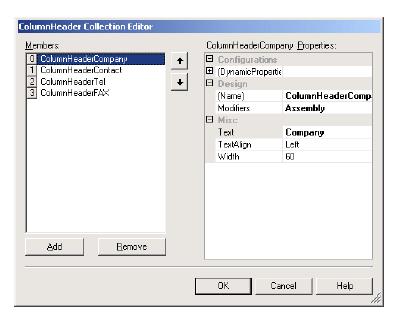
ولديك في مجلّد هذا الفصل تطبيق ListViewDemo، يمكنك التدرّب فيه على استخدام قائمة العرض.

**مجموعة الأعمدة Columns Collection:**

لعرض العناصر بتفاصيلها Report View، يجب عليك أولا أن تنشئ العدد المناسب من الأعمدة وتهيّئ خصائصها.. إنّ أوّل عمود يناظر العنصر، وباقي الأعمدة تناظر عناصره الفرعيّة (تفاصيله).

وللتعامل مع هذه الأعمدة، سنستخدم مجموعة الأعمدة Columns Collection.. إنّ كلّ عنصر في هذه المجموعة هو من النوع "عنوان العمود" ColumnHeader.

وأسهل طريقة لإنشاء هذه الأعمدة هو إنشاؤها في وقت التصميم، حيث يمنحك VB أداةً مرئيّةً لإنشائها.



افتح نافذة الخصائص وحدّد خاصية "الأعمدة" Columns.. اضغط زرّ الانتقال المقابل لها لتفتح نافذة محرّر عناوين مجموعة الأعمدة، حيث ستتمكّن بواسطتها من تحديد عدد الأعمدة وعناوينها.. كما يمكنك تحديد الحجم الذي سيظهر به كلّ عمود عند عرضه في القائمة.

ومن الممكن أيضا التعامل مع الأعمدة من خلال الكود، عن طريق الوسائل والخصائص المألوفة، ومنها:

**وسيلة الإضافة Add:**

هذه هي وسيلة الإضافة التقليديّة الموجودة في كلّ المجموعات، ولا جديد فيها سوى معاملاتها، حيث إنّ صيغتها كالتالي:

**CHeader = TreeView.Columns.Add(عنوان العمود, عرضه, محاذاة العنوان)**

وتعيد هذه الوسيلة كائنا من النوع ColumnHeader يمكنك من خلاله التعامل مع العمود الذي أضيف.. ويمتلك هذا الكائن ضمن صفاته خاصيّة اسم العمود Name، التي عن طريقها يمكنك منح كل عمود اسما، بحيث يمكنك الإشارة إليه من خلاله بدلا من الاكتفاء برقم العمود فحسب:

**Dim Header1 As ColumnHeader**

**Header1 = TreeView1.Columns.Add("Column 1", \_**

**60, ColAlignment.Left)**

**Header1.Name = "COL1"**

**وسيلة الحذف Remove:**

تمكّنك من حذف أيّ عمود.. والجملة التالية على سبيل المثال تحذف العمود الرابع:

**ListView1.Columns(3).Remove**

**ترتيب عناصر قائمة العرض:**

تمكّنك وسيلة الترتيب من ترتيب قائمة العرض تبعا لأيّ عمود من أعمدتها (لو كانت هناك أعمدة معروضة).

ونظرا لأنّ كلّ عمود يمكن أن يحتوي على أنواع مختلفةٍ من البيانات، فإنّ عليك أن تكتب دالة المقارنة بنفسك، تماما كما فعلنا من قبل مع المصفوفات والمجموعات في [الفصل التاسع](file:///C:\Users\Administrator\Desktop\02-%20Classes%20الخلايا\09-%20Collections%20خلايا%20المجموعات\2-%20واجهة%20العداد%20وواجهة%20المقارنة\3-%20الترتيب%20المخصّص.htm).

اتبع معنا هاتين الخطوتين وستجد الأمر غاية في البساطة إن شاء الله:

**الخطوة الأولى:**

اكتب الخليّة التي تنفّذ عمليّة المقارنة.. وفي المثال التالي تقوم هذه الخليّة بترتيب القائمة على أساس نصّ العنصر أي العمود الأوّل:

**Class ListSorter\_1**

**Implements IComparer**

**' دالة المقارنة**

**Public Function CompareTo(ByVal o1 As Object, ByVal o2 As \_**

**Object) As Integer \_**

**Implements System.Collections.IComparer.compare**

**Dim item1, item2 As String**

**item1 = CType(o1, ListViewItem).ToString.ToUpper**

**item2 = CType(o2, ListViewItem).ToString.ToUpper**

**If item1 > item2 Then**

**Return 1**

**ElseIf item1 < item2 Then**

**Return -1**

**Else**

**Return 0**

**End If**

**End Function**

**End Class**

أمّا هذه الخليّة فتعمل على ترتيب القائمة تبعا للرقم الذي يحتويه العنصر الفرعيّ الأوّل (العمود الثاني: رقم 1):

**Class ListSorter\_2**

**Implements IComparer**

**Public Function CompareTo(ByVal o1 As Object, ByVal o2 \_**

**As Object) As Integer \_**

**Implements System.collections.IComparer.compare**

**Dim item1, item2 As Integer**

**item1 = CInt (CType(o1, ListViewItem).SubItems(1).Text)**

**item2 = CInt(CType(o2, ListViewItem).SubItems(1).Text)**

**If item1 > item2 Then**

**Return 1**

**ElseIf item1 < item2 Then**

**Return -1**

**Else**

**Return 0**

**End If**

**End Function**

**End Class**

**الخطوة الثانية:**

حتّى تسمح للمستخدم بترتيب القائمة تبعا للعمود الذي يضغط عنوانه بالفأرة، استخدم حادث ضغط عنوان العمود ColumnClick، حيث يمكنك معرفة رقم العمود عن طريق الخاصيّة e.column، وبهذا يمكنك وضع اسم خليّة المقارنة المناسبة في خاصيّة "مرتّب عناصر القائمة" ListViewItemSorter، حتّى يتمّ الترتيب على أساسها.. هذا مع استخدام الوسيلة Sort لإجبار القائمة على ترتيب عناصرها تبعا لدالة المقارنة الخاصّة بك.. انظر المثال التالي:

**Public Sub ListView1\_ColumnClick(ByVal sender As Object, \_**

**ByVal e As System.WinForms.ColumnClickEventArgs) \_**

**Handles ListView1.ColumnClick**

**Select Case e.column**

**Case 0 ' العمود الأوّل**

**ListView1.ListViewItemSorter = New ListSorter\_1()**

**ListView1.Sort()**

**Case 1**

**ListView1.LisViewtItemSorter = New ListSorter\_2()**

**ListView1.Sort()**

**End Select**

**End Sub**

ويمكنك التدريب على هذه العملية في تطبيق ListViewDemo.

**التعامل مع العناصر المحدّدة في القائمة:**

نظرا لأنّ المستخدم يمكن أنّ يحدّد أكثر من عنصر معا، فإنّ لديك خاصيتين هامّتين تمكنانك من معرفة العناصر المحدّدة حاليّا في القائمة:

**مجموعة أرقام العناصر المحدّدة SelectedIndices Collection:**

تحتوي هذه المجموعة على أرقام العناصر المحدّدة حاليّا في القائمة، وهي تمدّك بالخصائص والوسائل التقليديّة للمجموعات، للتعامل مع هذه العناصر المحدّدة.

فمثلا العنصر الثاني في هذه المجموعة، رقمه في القائمة ينتج بالتعبير:

**ListView1.SelectedIndices(1)**

**مجموعة العناصر المحدّدة SelectedItems Collection:**

تحتوي هذه المجموعة على العناصر المحدّدة حاليّا في القائمة، وهي تمدّك بالخصائص والوسائل التقليديّة للمجموعات، للتعامل مع هذه العناصر المحدّدة.

والمثال التالي يريك كيف تطبع العناصر المحدّدة في القائمة:

**Private Sub bttnIterate\_Click(ByVal sender As System.Object, \_**

**ByVal e As System.EventArgs) Handles bttnIterate.Click**

**Dim LItem As ListViewItem**

**Dim I As Integer**

**For Each LItem In ListView1.SelectedItems**

**For I = 0 To LItem.SubItems.Count - 1**

**Console.Write(LItem.SubItems(I).Text & vbTab)**

**Next**

**Console.Write(vbCrLf)**

**Next**

**End Sub**

**تدريب:**

وأخيرا، لديك في مجلّد برامج هذا الفصل التطبيق CustomExplorer، وهو يستخدم الأداتين: الشجرة وقائمة العرض لإنشاء متصفّح لمحتويات القرص الصلب، شبيه بمتصفّح الويندوز Explorer.. استمتع بدراسته.. وبالمرّة خذ هذا التدريب: كتب مؤلّف المرجع هذا التطبيق ليحمّل في الشجرة عند بدء تشغيله، كلّ أسماء المجلّدات الفرعيّة، الموجودة في المجلّد C:\Program Files، وهي عمليّة غبيّة وتستهلك بعض الوقت.. لهذا السبب يقتصر التطبيق على استعراض محتويات المجلّد C:\Program Files، وليس سائر القرص الصلب.. هل لديك اقتراح أكثر عمليّة يحلّ هذه المشكلة، حتّى تتمكّن من عرض كلّ محتويات القرص الصلب قبل أن يقضي المستخدم نحبه؟

الإجابة:

* من الذكاء أن تكتفي بعرض أسماء المستوى الأوّل من عناصر الشجرة (أسماء محرّكات الأقراص Drives) فقط.
* في هذه الحالة لن تظهر علامة "+" بجوار اسم المجلّد، ممّا سيجعل المستخدم يظنّ أنّه لا يحتوى على أيّ مجلّدات فرعيّة.. ولحلّ هذه المشكلة، سنكتفي بإضافة مجلّد فرعيّ واحد لكلّ مجلّد لتظهر بجواره علامة "+".. (يمكنك أن تضيف كلّ المجلّدات الفرعيّة لمستوى واحد لو أردت، فهذا لن يستغرق وقتا يذكر).
* سنستخدم الخاصيّة الإضافيّة Tag لكلّ عنصر، حيث سنضع فيها القيمة صفر، لتدلّ على أنّ فرع المجلّد في الشجرة لم يكتمل ملؤه بمجلّداته الفرعيّة، والقيمة -1 إذا لم يكن المجلد الأصليّ على القرص الصلب يحتوى على مجلّدات فرعيّة بطبيعة الحال، والقيمة 1 لندلّ أنّ فرع المجلّد قد تمّ ملؤه بكلّ مجلّداته الفرعيّة.
* وعندما يحاول المستخدم إسدال مجلّدات أيّ فرع في الشجرة، يمكنك استخدام الحدث BeforeExpand، فإذا كانت قيمة الخاصيّة Tag لهذا الفرع = 0، فستقوم باستكمال إضافة المجلّدات الفرعيّة لهذا المجلّد إلى الشجرة، وستضيف لكلّ مجلّد منها واحدا فقط من مجلّداته الفرعيّة، مع وضع القيمة المناسبة لخاصيّة Tag لكلّ منها.

بهذا لن يشعر المستخدم بمرور أيّ وقت، مهما كان عدد المجلّدات الموجودة على القرص الصلب!.. جرّب كتابة هذا الكود.. وعلى كلّ حال، ستجده في تطبيق DragAndDrop في مجلّد برامج هذا الفصل.

**حفظ محتويات قائمة العرض في ملفّ:**

لتجدنّ هذا الأمر أسهل بمراحل من حفظ عناصر الشجرة، فوضع العناصر والعناصر الفرعيّة في المصفوفة القائمة ArrayList أسهل بكثير.. فقط عليك أن تراعي أن تعرّف سجلا ملائما لتركيب القائمة وعناصرها.. حاول أن تجرّب ذلك.

**السحبُ والإدراج**

* [**مشروع السحب والإدراج:**](file:///C:\Users\Administrator\Desktop\vb.net\d-%20%20مواضيع%20متقدّمة\11-%20أدوات%20متقدّمة\5-%20السحبُ%20والإدراج\1-%20مشروع%20السحب%20والإدراج.htm)
* [**مربّع النصّ المنسّق:**](file:///C:\Users\Administrator\Desktop\vb.net\d-%20%20مواضيع%20متقدّمة\11-%20أدوات%20متقدّمة\5-%20السحبُ%20والإدراج\2-%20مربّع%20النصّ%20المنسّق.htm)
* [**قائمة العرض:**](file:///C:\Users\Administrator\Desktop\vb.net\d-%20%20مواضيع%20متقدّمة\11-%20أدوات%20متقدّمة\5-%20السحبُ%20والإدراج\3-%20قائمة%20العرض.htm)
* [**الشجرة:**](file:///C:\Users\Administrator\Desktop\vb.net\d-%20%20مواضيع%20متقدّمة\11-%20أدوات%20متقدّمة\5-%20السحبُ%20والإدراج\4-%20الشجرة.htm)

**مشروع السحب والإدراج:**

أعرف أنّك ستزفر قائلا في استياء:

* آه.. عدنا إلى تلك المتاهة.

إذ لا بدّ أنّك ما تذكر المشاقّ التي واجهناها في الفصل السادس.

اطمئنّ.. الموضوع هنا أسهل بكثير، فمربّع النصّ المنسّق والشجرة وقائمة العرض تسهّل لك القيام بعمليّة السحب والإدراج بشكل كبير.. تعالَ نرَ:

ابدأ تطبيقا جديدا، وأسمه DragAndDrop.. صمّم واجهة استخدامه كما في الصورة التالية:



تلاحظ طبعا أنّ الشجرة تعرض محتويات القرص الصلب والأسطوانة والقرص المرن، وأنّ قائمة العرض تعرض محتويات المجلّد المحدّد في الشجرة.. لن أكتب الكود الذي يفعل ذلك هنا.. سأترك لك استكشافه بنفسك في التطبيق، مستعينا بالخوارزميّة التي أوضحناها سابقا في الفقرة قبل السابقة.

ما يعنينا الآن هو السحب والإدراج.. نريد أن نسحب العناصر بين الأدوات الثلاثة: مربّع النصّ المنسّق (وسنسمّيه Rt)، والشجرة ( وسنسمّيها T)، وقائمة العرض (وسنسمّيها L)، تبعا للشروط التالية:

* عند سحب نصّ إلى الشجرة، يتمّ إضافته كعنصر فرعيّ تحت جذر "العناصر المسقطة".
* عند سحب فرع من الشجرة وإسقاطه عليها، يتمّ نسخه وكلّ ما يحتويه من عناصر إلى جذر "العناصر المسقطة".
* عند سحب فرع من الشجرة وإسقاطه على مربّع النصّ المنسّق، يتمّ كتابة أسماء كلّ عنصر وما يندرج تحته من عناصر فرعيّة في مربّع النصّ المنسّق، بحيث يكون كلّ عنصر في سطر منفرد.
* عند السحب من أو الإسقاط على قائمة العرض، نتعامل مع اسم العنصر كنصّ فقط، وليس كفرع في الشجرة.

تعالَ ننفّذ ذلك:

**مربّع النصّ المنسّق:**

لحسن الحظّ لن نقوم بذلك اللفّ والدوران الذي اتبعناه مع مربّع النصّ العاديّ.. كل ما سنفعله، هو أن نبدأ عمليّة السحب في حدث ضغط الفأرة MouseDown (أو حدث حركة الفأرة MouseMove لو أردت) كالتالي:

**Private Sub Rt\_MouseDown(ByVal sender As Object, ByVal e As \_**

**MouseEventArgs) Handles Rt.MouseDown**

**If e.Button = MouseButtons.Left Then**

**If Rt.SelectedText <> "" Then**

**Rt.DoDragDrop(Rt.SelectedText, \_**

**DragDropEffects.Copy)**

**End If**

**End If**

**End Sub**

باديةٌ طبعا بساطة العمليّة.

الآن علينا أن نستخدم الحدث DragEnter لتحديد التأثير الذي سيظهر عندما يدخل العنصر المسحوب المجال الجوّيّ لمربّع النصّ المنسّق.. فقط سنتأكّد من أنّ نوع البيانات المسحوبة نصّ أو فرع شجرة TreeNode.. ما عدا هذا لن نسمح بالإسقاط.. ونظرا لأنّ كود هذا الحدث لن يختلف في الشجرة أو قائمة العرض، فقد جعلناه يستجيب للأدوات الثلاثة معا:

**Private Sub TheDragEnter(ByVal sender As Object, ByVal e As \_**

**DragEventArgs) Handles T.DragEnter, L.DragEnter, \_**

**Rt.DragEnter**

**If e.Data.GetDataPresent(GetType(TreeNode)) OrElse \_**

**e.Data.GetDataPresent(DataFormats.Text) Then**

**e.Effect = DragDropEffects.Copy**

**Else**

**e.Effect = DragDropEffects.None**

**End If**

**End Sub**

**ملحوظة:**

لسبب لا أدريه، لا يعرض مربّع النصّ المنسّق أسماء الأحداث الخاصّة بالسحب والإدراج ضمن الأحداث الخاصّة به في القائمة العلويّة اليمنى في محرّر الكود.. ولكنّ هذا لا يعني أنّ هذه الأحداث غير موجودة.. فقط تحامل على نفسك، واكتب صيغتها بنفسك، أو انسخها من هذا الفصل!

الآن لم يبقَ إلا نكتب كود الإسقاط فوق مربّع النصّ.. طبعا سنحتاج لإجراء ارتداديّ Recursive في حالة إسقاط فرع من الشجرة، وذلك حتّى نمرّ عبر كلّ عناصر هذا الفرع.. تعال نبدأ بكتابة هذا الإجراء، مع ملاحظة أنّنا كلّما توغّلنا لمستوى أعمق داخل مستويات فرع الشجرة، سنزيد من هامش السطر في مربّع النصّ المنسّق (باستخدام خاصيّة SelectionIndent)، وذلك حتّى يتمّ عرض أسماء العناصر بشكل منسّق:

**Sub TraceNodes(ByRef N As TreeNode)**

**Dim C, Idnt As Integer**

**' يجب الاحتفاظ بالهامش في متغيّر للاحتفاظ به لحين الصعود من المستويات الأعمق**

**Idnt = Rt.SelectionIndent + 16**

**For C = 0 To N.GetNodeCount(False) - 1**

**Rt.SelectedText = vbLf & N.Nodes(C).Text**

**' زيادة هامش عناصر هذا المستوى**

**Rt.SelectionIndent = Idnt**

**' المرور عبر العناصر الفرعيّة للعنصر الحاليّ**

**TraceNodes(N.Nodes(C))**

**Next**

**End Sub**

تعالَ الآن نرى كود الإسقاط:

**Private Sub Rt\_DragDrop(ByVal sender As Object, ByVal e As \_**

**DragEventArgs) Handles Rt.DragDrop**

**If e.Data.GetDataPresent(GetType(TreeNode)) Then**

**' إسقاط فرع من الشجرة**

**' تغيير موضع الكتابة إلى الموضع الذي سيتمّ الإسقاط عليه**

**' لاحظ أنّنا نحتاج أولا لتحويل النقطة**

**' من إحداثيّات الشاشة إلى إحداثيّات مربّع النصّ المنسّق**

**Rt.SelectionStart = Rt.GetCharIndexFromPosition \_**

**(Rt.PointToClient(New Point(e.X, e.Y)))**

**Dim N As TreeNode**

**N = CType(e.Data.GetData(GetType(TreeNode)), TreeNode)**

**' كتابة اسم العنصر**

**Rt.SelectedText += N.Text**

**' كتابة كلّ العناصر الفرعيّة لهذا العنصر**

**TraceNodes(N)**

**ElseIf e.Data.GetDataPresent(DataFormats.Text) Then ' إسقاط نصّ**

**Rt.SelectionStart = Rt.GetCharIndexFromPosition \_**

**(Rt.PointToClient(New Point(e.X, e.Y)))**

**Rt.SelectedText += e.Data.GetData(DataFormats.Text)**

**End If**

**End Sub**

**قائمة العرض:**

ما زالت هناك مشكلة في السحب من مربّع النصّ المنسّق.. إنّ إطلاق عمليّة السحب في الحدث MouseDown سيؤدّي إلى حدوث سحب وإسقاط في مربّع النصّ كلّما نقر المستخدم أيّ نصّ مرّتين بزرّ الفأرة الأيسر Double Click!

ولو كتبت كود السحب والإدراج في حدث حركة الفأرة، فستبدأ عمليّة السحب قبل أن ينتهي المستخدم من تحديد كلّ النصّ الذي يريد تحديده بالفأرة!

وطبعا عليك بالتفكير في حلّ لهذه المشكلة، التي لن تواجهنا لحسن الحظّ في عمليّة السحب من قائمة العرض، وذلك لأنّها تمتلك الحدث "سحب العنصر" ItemDrag، الذي ينطلق توَّ أن تشرع في سحب أيّ عنصر من القائمة.

معنى هذا أنّ عمليّة السحب ستبدأ بالكود البسيط التالي:

**Private Sub L\_ItemDrag(ByVal sender As Object, ByVal e As \_**

**ItemDragEventArgs) Handles L.ItemDrag**

**L.DoDragDrop(CType(e.Item, ListViewItem).Text, \_**

**DragDropEffects.Copy)**

**End Sub**

لم يبقَ إلا أن نكتب كود الإسقاط، وهو أبسط من ذلك الخاصّ بمربّع النصّ المنسّق:

**Private Sub L\_DragDrop(ByVal sender As Object, ByVal e As \_**

**DragEventArgs) Handles L.DragDrop**

**If e.Data.GetDataPresent(GetType(TreeNode)) Then**

**Dim N As TreeNode**

**N = CType(e.Data.GetData(GetType(TreeNode)), TreeNode)**

**L.Items.Add(N.Text)**

**L.EnsureVisible(L.Items.Count - 1)**

**ElseIf e.Data.GetDataPresent(DataFormats.Text) Then**

**L.Items.Add(e.Data.GetData(DataFormats.Text))**

**L.EnsureVisible(L.Items.Count - 1)**

**End If**

**End Sub**

**ملحوظة:**

يمكن إسقاط العنصر في موضع الفأرة، بدلا من إضافته لنهاية قائمة العرض، وذلك باستخدام الكود التالي بدلا من السطر الذي يستخدم الوسيلة Add:

**Dim P As Point = L.PointToClient(New Point(e.X, e.Y))**

**Dim I As Integer = L.GetItemAt(P.X, P.Y).Index**

**L.Items.Insert( I, N.Text)**

**الشجرة:**

تمتلك الشجرة بدورها الحدث ItemDrag، ممّا يعنى أنّ كود السحب سيكون بسيطا.. الجديد، هو استخدامنا للوسيلة "استنساخ" Clone لنسخ الفرع المسحوب من الشجرة، وذلك لأنّنا لا نستطيع وضع نفس الفرع مرّتين في الشجرة، إلا باستخدام هذه الطريقة:

**Private Sub T\_ItemDrag(ByVal sender As Object, ByVal e As \_**

**ItemDragEventArgs) Handles T.ItemDrag**

**T.DoDragDrop(CType(e.Item, TreeNode).Clone, \_**

**DragDropEffects.Copy)**

**End Sub**

وأخيرا، ها هو ذا كود الإسقاط.. لاحظ أنّنا سنضع العناصر المسقطة في فرع خاصّ بها يسمّى ("العناصر المسقطة"):

**Private Sub T\_DragDrop(ByVal sender As Object, ByVal e As \_**

**DragEventArgs) Handles T.DragDrop**

**If e.Data.GetDataPresent(GetType(TreeNode)) Then**

**T.Nodes(2).Nodes.Add(e.Data.GetData(GetType(TreeNode)))**

**T.Nodes(2).Expand()**

**ElseIf e.Data.GetDataPresent(DataFormats.Text) Then**

**T.Nodes(2).Nodes.Add(e.Data.GetData(DataFormats.Text))**

**T.Nodes(2).Expand()**

**End If**

**End Sub**

ولكن لماذا وضعنا العناصر المسقطة في موضع خاصّ بها، ولم نضفها للشجرة في موضع الإسقاط؟

ببساطة، حتّى لا نفسد عمل الشجرة، كوسيلة لتصفّح محتويات الكمبيوتر.. ولكن لو كنت مصرّا، يمكنك أن نضيف العنصر للشجرة في موضع إسقاطه بالكود التالي:

**Dim N As TreeNode**

**N = T.GetNodeAt(T.PointToClient(New Point(e.X, e.Y)))**

**If N Is Nothing Then ' الإسقاط في موضع خالٍ من الشجرة**

**T.Nodes.Add(e.Data.GetData(GetType(TreeNode)))**

**ElseIf N.Parent Is Nothing Then ' الإسقاط كجذر من جذور الشجرة**

**T.Nodes.Insert(N.Index, e.Data.GetData(GetType(TreeNode)))**

**Else ' الإسقاط كعنصر فرعيّ**

**N.Parent.Nodes.Insert(N.Index, \_**

**e.Data.GetData(GetType(TreeNode)))**

**End If**

وهذا يجرّنا لكي نتعرّف على كيفيّة إسدال فروع الشجرة، لو حلّقنا فوقها لوهلة بالعنصر المسحوب، وذلك حتّى نسمح للمستخدم بتحديد الموضع الذي يريد الإسقاط فيه في الشجرة بسهولة.

هنا سنستخدم الحدث GiveFeedback، الذي يحدث على فترات أثناء عمليّة السحب:

**Private Sub T\_GiveFeedback(ByVal sender As Object, ByVal e As \_**

**GiveFeedbackEventArgs) Handles T.GiveFeedback**

**Dim N As TreeNode**

**N = T.GetNodeAt(T.PointToClient(Me.MousePosition))**

**If Not N Is Nothing Then**

**N.Expand()**

**N.EnsureVisible()**

**End If**

**End Sub**

هناك مشكلة صغيرةٌ ستقابلك، وهي أنّ الفروع ستنسدل بسرعة كبيرة، نظرا لأنّ هذا الحدث يستدعى على فتراتٍ زمنيّة غايةٍ في الصغر.. هل تريد حلا؟

حسنا.. لن نسدل العناصر في كلّ مرّة ينطلق فيها الحدث GiveFeedback، بل سنستخدم المتغيّرات الزمنيّة لنحسب مرور 0.8 من الثانية على استمرار تحليق الفأرة فوق أحد العناصر.. فإذا مرّت هذه المدّة دون أن تتحرّك الفأرة، نسدل هذا العنصر.. هذا هو الكود الذي يقوم بذلك.. لاحظ أنّه يستجيب للأدوات الثلاثة، وذلك لمراعاة حالتي السحب من مربّع النصّ وقائمة العرض إلى الشجرة:

**Private Sub T\_GiveFeedback(ByVal sender As Object, ByVal e As \_**

**GiveFeedbackEventArgs) Handles \_**

**T.GiveFeedback, L.GiveFeedback, Rt.GiveFeedback**

**Static LastN As TreeNode, STime As Date = Now**

**Static Started As Boolean**

**Dim N As TreeNode**

**N = T.GetNodeAt(T.PointToClient(Me.MousePosition))**

**' إذا لم تكن الفأرة فوق أحد فروع الشجرة، فغادر الإجراء**

**If N Is Nothing Then**

**' ألغِ عمليّة حساب الوقت**

**Started = False**

**Exit Sub**

**End If**

**If N Is LastN Then ' الفأرة ما زالت فوق نفس العنصر**

**If Started Then ' عمليّة حساب الوقت بدأت فيما سبق**

**' احسب الوقت المنقضي منذ بدأت عمليّة حساب الوقت**

**Dim Ts As TimeSpan = Now.Subtract(STime)**

**If Ts.TotalSeconds >= 0.8 Then**

**' لا تنس إلغاء عمليّة حساب الوقت**

**Started = False**

**N.Expand()**

**N.EnsureVisible()**

**End If**

**Else ' عمليّة حساب الوقت يجب أن تبدأ الآن**

**Started = True**

**STime = Now**

**End If**

**Else ' الفأرة لم تعد فوق نفس العنصر**

**Started = False ' ألغِ عمليّة حساب الوقت**

**End If**

**'العنصر الحالي سيصبح هو "العنصر السابق" عند انطلاق هذا الحدث في المرّة القادمة**

**LastN = N**

**End Sub**

1. **بالاستفادة من موضوع عن تقنية المرآة، لعاشق البرمجة (أحمد عبد الرازق).** [↑](#footnote-ref-1)