**الفصل الثامن**

**التعامل مع النصوص والتواريخ**

* [**التعامل مع الحروف والنصوص:**](file:///C:\Users\Administrator\Desktop\vb.net\c-%20Classes%20الخلايا\08-%20خلايا%20النصوص%20والتواريخ\08-%20خلايا%20النصوص%20والتواريخ\1-%20التعامل%20مع%20الحروف%20والنصوص.htm)
* [**التعامل مع التواريخ:**](file:///C:\Users\Administrator\Desktop\vb.net\c-%20Classes%20الخلايا\08-%20خلايا%20النصوص%20والتواريخ\08-%20خلايا%20النصوص%20والتواريخ\2-%20التعامل%20مع%20التواريخ.htm)

**التعامل مع الحروف والنصوص**

للتعامل مع النصوص، يمنحك إطار العمل .Net Frame Work خليتين لذلك: "النص" String و"باني النص" StringBuilder.. هذا بالإضافة إلى خليّة الحروف Char، للتعامل مع الحروف المنفردة.

* [**خليّة الحروف Char Class:**](file:///C:\Users\Administrator\Desktop\vb.net\c-%20Classes%20الخلايا\08-%20خلايا%20النصوص%20والتواريخ\1-%20التعامل%20مع%20الحروف%20والنصوص\1-%20خليّة%20الحروف.htm)
* [**خليّة النصوص String Class:**](file:///C:\Users\Administrator\Desktop\vb.net\c-%20Classes%20الخلايا\08-%20خلايا%20النصوص%20والتواريخ\1-%20التعامل%20مع%20الحروف%20والنصوص\2-%20خليّة%20النصوص.htm)
* [**خليّة باني النصّ StringBuilder Class:**](file:///C:\Users\Administrator\Desktop\vb.net\c-%20Classes%20الخلايا\08-%20خلايا%20النصوص%20والتواريخ\1-%20التعامل%20مع%20الحروف%20والنصوص\3-%20خليّة%20باني%20النصّ.htm)

**خليّة الحروف Char Class**

* [**وظيفتها:**](file:///C:\Users\Administrator\Desktop\vb.net\c-%20Classes%20الخلايا\08-%20خلايا%20النصوص%20والتواريخ\1-%20خليّة%20الحروف\1-%20وظيفتها.htm)
* [**الوسائل:**](file:///C:\Users\Administrator\Desktop\vb.net\c-%20Classes%20الخلايا\08-%20خلايا%20النصوص%20والتواريخ\1-%20خليّة%20الحروف\2-%20الوسائل.htm)
* [**الدوالّ الحرفيّة في VB:**](file:///C:\Users\Administrator\Desktop\vb.net\c-%20Classes%20الخلايا\08-%20خلايا%20النصوص%20والتواريخ\1-%20خليّة%20الحروف\3-%20الدوالّ%20الحرفيّة.htm)

**وظيفتها**:

يتمّ تخزين الحرف الواحد في هذه الخليّة في وحدتي ذاكرة 2 Bytes.

**ملحوظة:**

يعبّر ترميز ASCII عن 256 حرفا فقط، لذلك يتمّ تمثيل الحرف فيه بوحدة ذاكرة Byte واحدة فقط.. أمّا ترميز Unicode، فهو يعبّر عن عدد موسّع من الحروف، لذلك يتمّ تمثيل الحرف فيه بوحدتي ذاكرة 2 Bytes.

ويمكنك تعريف المتغيرات من هذا النوع كالتالي:

**Dim ch As Char = "A"c**

فإذا كنّا نضع حروف النصّ بينَ علامتي تنصيص ""، فإنّ الحرف يُعامل بالمثل ولكن مع وضع حرف "c" بعد علامتي التنصيص.. أو يمكنك أن تستخدم دالة التحويل إلى حرف CChar، لتحويل النصّ إلى حرف كالتالي:

**Dim ch As Char = CChar("A")**

**وسائل هذه الخليّة:**

للدوال التي سنناقشها في هذا المقطع صيغتان:

- الصيغة الأولى تقبل معاملا واحدا، هو الحرف الذي ستعمل عليه:

**Char.IsDigit(حرف)**

- والصيغة الثانية تقبل معاملين، أحدهما نصّ، والآخر موضع الحرف المطلوب التعامل معه في هذا النصّ:

**Char.IsDigit(نصّ, موضع الحرف في النصّ)**

أعتقد أنّك لن تكون بحاجة لتذكيرك بهاتين الصيغتين في كل الدوال التالية.

تعالَ نبدأ بالدالة الشيّقة التالية:

**اقرإ الطائفة GetUnicodeCategory:**

تعيد لك هذه الدالة الطائفة التي ينتمي إليها الحرف.. هذه الطائفة هي تعبير رقميّ من أعضاء المرقّم UnicodeCategory، التي هي كالتالي:

التعبيرات الدالة على أنّ الحرف من علامات الترقيم:

|  |  |
| --- | --- |
| EnclosingMark | الحرف من علامات الاحتواء (كالأقواس). |
| OpenPunctuation | علامة الترقيم الأولى في زوج من العلامات مثل الأقواس. |
| **ClosePunctuation** | علامة الترقيم الثانية في زوج من العلامات مثل الأقواس. |
| ConnectorPunctuation | الحرف من العلامات التي تصل بين حرفين مثل "ـ". |
| DashPunctuation | الحرف هو علامة الترقيم "\_". |
| InitialQuotePunctuation | علامة التنصيص الأولى. |
| FinalQuotePunctuation | علامة التنصيص الثانية (الأخيرة). |
| OtherPunctuation | علامة ترقيم أخرى غير ما سبق. |

التعبيرات الدالّة على أنّ الحرف يمثّل رقما:

|  |  |
| --- | --- |
| DecimalDigitNumber | الحرف يمثّل رقما عشريا. |
| LetterNumber | حرف يمثّل رقما.. وهو يختلف عن الأرقام من 0 إلى 9.. فمثلا: الرقم 5 يكتب باللاتينية V. |
| OtherNumber | الحرف يمثّل رقما آخر غير النوعين السابقين. |

التعبيرات الدالّة على أنّ الحرف حرف تنسيق وتحكّم:

|  |  |
| --- | --- |
| Control | حرف تحكّم |
| Format | الحرف من علامات التنسيق التي لا تظهر في النصّ المكتوب. |
| LineSeparator | فاصل أسطر (حرف إدخال Enter). |
| ParagraphSeparator | فاصل مقاطع. |
| SpaceSeparator | مسافة فاصلة. |
| SpacingCombiningMark | علامة دمج فاصلة. |
| NonSpacingMark | ليس علامة من علامات المسافة. |
| Surrogate | حرف استبدال علوي أو سفلي. |
| TitlecaseLetter | حرف من حروف عنوان. |

التعبيرات الدالّة على نوع الرمز الذي يمثّله الحرف:

|  |  |
| --- | --- |
| ModifierLetter | حرف مستقلّ يؤثّر في السابق له. |
| ModifierSymbol | رمز تعديل، مثل علامة القسمة، التي تدلّ أن ما قبلها هو البسط وما يليها هو المقام. |
| CurrencySymbol | علامة العملة |
| UppercaseLetter | حرف كبير Capital. |
| LowercaseLetter | حرف صغير Small. |
| MathSymbol | رمز رياضيّ، مثل +، -، \*، / .. إلخ. |
| OtherSymbol | رمز آخر غير ما سبق. |

أي حالة أخرى:

|  |  |
| --- | --- |
| OtherLetter | حرف آخر غير ما مضى. |
| PrivateUse | حرف مخصّص لاستخدام خاصّ. |
| OtherNotAssigned | حرف ليس له طائفة مُعرّفة. |

**اقرإ القيمة الرقميّة GetNumericValue:**

لو أرسلت حرفا يمثّل رقما بين 0 و 9 كمعامل لهذه الدالة، فإنّها ترجع لك هذا الرقم (كقيمة رقميّة هذه المرّة وليس كحرف)، ولو كان المعامل أيّ حرف غير رقميّ، ترجع هذه الدالة -1.

**إنّه حرف تحكّم IsControl:**

ترجع هذه الدالة True إذا كان الحرف المضغوط من حروف التحكّم، مثل Backspace، و Escape... إلخ.

**إنّه رقم IsDigit:**

ترجع هذه الدالة True إذا كان الحرف رقما بين 0 و 9.

والإجراء التالي هو المستجيب لحدث ضغط أزرار لوحة المفاتيح، وفيه ترى كيف يمكن استخدام هذه الوسيلة هي وسابقتها للسماح بكتابة الأرقام فقط في مربّع النصّ:

**Private Sub TextBox1\_KeyPress(ByVal sender As Object, ByVal e \_**

**As EventArgs) Handles TextBox1.KeyPress**

**Dim c As Char**

**c = e.KeyChar**

**If Not (Char.IsDigit(c) Or Char.IsControl(c)) Then**

**e.Handled = True ' إلغاء الحرف المضغوط**

**End If**

**End Sub**

وطبعا لاحظت أنّنا سمحنا للمستخدم باستخدام حروف التحكّم، حيث إنّه سيحتاج لمسح أو تعديل الأرقام.. ولو كان المستخدم سيدخل أرقاما عشريّة، فيجب أن تسمح له بكتابة العلامة العشريّة، كالتالي:

**If Not (Char.IsDigit(c) Or c = "."c Or Char.IsControl(c)) Then**

**e.Handled = True**

**End If**

لاحظ أنّ الكود السابق لا يستطيع أن يمنع المستخدم من كتابة أكثر من علامة عشريّة في نفس الرقم!.. وسأترك لك محاولة معالجة هذا الأمر كتدريب.

**إنّه حرف IsLetter:**

ترجع هذه الدالة True إذا كان الحرف المضغوط من حروف الكتابة.

**إنّه حرف أو رقم IsLetterOrDigit:**

ترجع هذه الدالة True إذا كان الحرف المضغوط رقما أو حرفا.

**إنّه صغير IsLower:**

ترجع هذه الدالة True إذا كان الحرف المضغوط صغيرا Small.

**إنّه كبير IsUpper:**

ترجع هذه الدالة True إذا كان الحرف المضغوط كبيرا Capital.

**إنّه عدد IsNumber:**

مماثلة للدالة IsDigit، إلا إنّها تأخذ في اعتبارها الحروف الداخلة في تمثيل الأرقام في النظام السداسي عشري، وهي الأرقام التالية للرقم 9، حيث نمثّلها بالحروف: A و B و C و D و E و F، وهي تناظر الأعداد من 10 إلى 15 في النظام العشريّ.

**إنّه علامة ترقيم IsPunctuation:**

أعتقد أنّ وظيفتها واضحة!

**إنّه فاصل IsSeparator:**

ترجع هذه الدالة True إذا كان الحرف المضغوط من الحروف الفاصلة، مثل المسافة وحرف الإدخال Enter ، والفاصلة والنقطتان المتعامدتان والأقواس ونقطة التوقّف ... إلخ.

**إنّه مسافة بيضاء IsWhiteSpace:**

ترجع هذه الدالة True إذا كان الحرف المضغوط مجموعة مسافات أو حرف الجدولة Tab أو مغذيات السطر Line feeds... إلخ.

**أمّا الوسائل الثلاث التالية فهي غنيّة عن التعريف:**

حوّل إلى حرف صغير **ToLower**، حوّل إلى حرف كبير **ToUpper**، حوّل إلى نصّ **ToString**.

**الدوالّ الحرفيّة في VB:**

بالإضافة لوسائل خليّة الحروف، يمنحك VB الدوال التالية:

**Asc:**

تعيد الكود الرقميّ في ترميز ASCII، والذي يمثّل الحرف الأوّل في النصّ المرسل لها كمعامل.

**AscW:**

تعيد الكود الرقميّ في ترميز Unicode، والذي يمثّل الحرف الأوّل في النصّ المرسل لها كمعامل.. فإذا لم يكن نظام التشغيل يتعامل مع الحروف من النوع Unicode، فإنّ هذه الدالة تتصرّف مثل الدالة السابقة.

**Chr:**

تقبل هذه الدالة رقما، وتعيد لك الحرف الذي يمثّله في ترميز ASCII.

**ChrW:**

تقبل هذه الدالة رقما، وتعيد لك الحرف الذي يمثّله في ترميز Unicode.

**CChr:**

تقبل هذه الدالة متغيّرا من أيّ نوعٍ، وتعيد لك متغيّرا حرفيّا يمثّل القيمة التي بالمتغيّر الأصليّ ـ إن كانت قابلة للتحويل لحرف.

**خليّة النصوص String Class**

* [**الخصائص:**](file:///C:\Users\Administrator\Desktop\vb.net\c-%20Classes%20الخلايا\08-%20خلايا%20النصوص%20والتواريخ\2-%20خليّة%20النصوص\1-%20الخصائص.htm)
* [**الوسائل:**](file:///C:\Users\Administrator\Desktop\vb.net\c-%20Classes%20الخلايا\08-%20خلايا%20النصوص%20والتواريخ\2-%20خليّة%20النصوص\2-%20الوسائل.htm)
* [**الدوال النصّيّة في VB:**](file:///C:\Users\Administrator\Desktop\vb.net\c-%20Classes%20الخلايا\08-%20خلايا%20النصوص%20والتواريخ\2-%20خليّة%20النصوص\3-%20الدوال%20النصّيّة.htm)

**خصائص خليّة النصوص:**

**الطول Length:**

تعيد لك هذه الخاصيّة عدد حروف النصّ الموجود في المتغيّر الحالي:

**Dim L As Integer = Word.Length**

**حروف Chars:**

تعتبر هذه الخاصيّة مصفوفة، تسمح لك بالتعامل مع حروف النصّ على حسب أرقام خاناتها.. وكأيّ مصفوفة، فإنّ أوّل خانة تحمل الرقم 0.. وللأسف تسمح لك هذه الخاصيّة بقراءة أيّ حرف من النصّ ولكن لا تسمح لك بتغييره.

**وسائل خليّة النصوص:**

**استبدال Replace:**

تمكّنك هذه الوسيلة من تغيير أيّ عددٍ من حروف النصّ، سواء بوضع حروف أخرى مكانها، أو بوضع نصّ فارغ "" (فكأنّك قد قمت بحذف هذه الحروف).

وهذه الوسيلة لا تؤثّر على المتغيّر الذي يحتوي على النصّ الأصليّ، وإنّما تعيد لك نصّا جديدا بعد إجراء التغييرات عليه.. فإذا كان من المحتّم أن تغيّر النصّ الأصليّ، فضع النتيجة في المتغيّر الذي يحتويه.

**Dim Word As String = "استبدال"**

**Word = Word.Replace("ال", "ل")**

**نسخ Copy:**

هذه الوسيلة مشتركة Shared، بمعنى أنّها تنطبق على الخليّة الأصليّة وليس على نسخة بعينها من الخليّة.. لهذا فإنّ هذه الوسيلة تتطلّب إرسال النصّ كمعامل لها، حيث تقوم بنسخه إلى متغيّر آخر، الأمر الذي يمكن أن تقوم به بعمليّة "=" مباشرةً:

**Dim s1, s2 As String**

**s1 = "This Is a String"**

**s2 = String.Copy(s1)**

**مقارنة Compare:**

تقارن بين نصيّن، فإذا كان أوّلهما أصغر (في ترتيبه الهجائيّ) كان ناتجها -1، وإذا كانا متساويين كان ناتجها صفرا، وإذا كان أوّلهما أكبر كان ناتجها 1.. وهي على الصيغة التالية:

**String.Compare(النصّ الأوّل, النصّ الثاني)**

ويمكن أن تكون على الصيغة التالية:

**String.Compare(النصّ الأوّل, النصّ الثاني, مراعاة حالة الحروف)**

حيث يمكن أن يكون المعامل الثالث True لمراعاة حالة الحروف عند المقارنة، أو False لعدم مراعاتها.

وهناك صيغة ثالثة، تسمح لك بمقارنة جزء من النصّيّن:

**String.Compare(النصّ الأوّل, بدايته, النصّ الثاني, بدايته, طول الجزء المقارن)**

أمّا الصيغة الرابعة فهي مماثلة للثالثة، مع إضافة معامل رابع لاعتبار حالة الأحرف أو عدمه.

**تشبيك Concat:**

يمكنك استخدام هذه الوسيلة للحْمِِ نصّين في بعضهما لإنتاج نصّ جديد.. وأبسط صيغة لهذه الدالة هي:

**newString = String.Concat(Str1, Str2)**

وهي بهذا مكافئة لعمليّة جمع النصوص كالتالي:

**newString = Str1 + Str2**

أو كالتالي:

**newString = Str1 & Str2**

ويمكن أن ترسل لهذه الدالة أيّ عدد من النصوص لتشبيكها معا، وذلك لأنّ إحدى صيغها تقبل مصفوفة معاملات ParamArray، لهذا فإنّ تعبيرا كالتالي يمكن استخدامه:

**newString = String.Concat(Str1, Str2, Str3, Str4, Str5, Str6, Str7)**

أمّا الصيغة الأهمّ، فهي التي تقبل كمعامل لها كائنا Object.. وبما أنّ المصفوفة كائن، فمن الممكن أن ترسل مصفوفة من النصوص كمعامل لهذه الدالة، لتشبيك كل النصوص الموجودة في خاناتها لإنتاج نصّ واحد:

**Dim strings() As String = {"str1", "str2", "str3", "str4"}**

**Dim longString As String**

**longString = String.Concat(strings)**

وبعد تنفيذ الجملة الأخيرة ستصبح قيمة المتغيّر longString كالتالي:

str1str2str3str4

**توصيل Join:**

هذه الدالة مماثلة لدالة التشبيك Concat، فهي أيضا تقوم بتشبيك النصوص الموجودة في خانات مصفوفة لتنتج نصا واحدا جديدا، ولكنّها تختلف في أنّها تسمح لك بوضع حرف فاصل بين النصوص عند تشبيكها، مثل المسافة أو الفاصلة أو أيّ نصّ آخر.

**newString = String.Join(نصّ فاصل, مصفوفة نصوص)**

وإذا كنت ترغب في توصيل بعض عناصر المصفوفة فقط، فاستخدم الصيغة التالية:

**newStr = String.Join(نصّ فاصل, مصفوفة نصوص, خانة البداية, عدد الخانات)**

**انشطار Split:**

هذه هي الدالة العكسيّة للدالة السابقة، فلو طبّقتها على النصّ الذي تم تكوينه من دالة التوصيل، فستحصل على مصفوفة النصوص مرّة أخرى.. إذن فهذه الدالة تقسّم النصّ إلى أجزاء صغيرة وتضعها في مصفوفة، بناءً على وجود حرف فاصل أو أكثر بين أجزاء النصّ، مثل المسافة أو الفاصلة.

**strings() = myString.Split(مصفوفة معاملات من الحروف الفاصلة)**

قديما عندما كنت أستخدم VB5 كنت أضطرّ لكتابة برنامج طويل لتقسيم النصّ إلى مجموعة من الأسطر وأقوم بحفظها في مصفوفة، بعد ذلك كنت أمرّ على كلّ المصفوفة سطرا سطرا لتقسيم كل سطر إلى مصفوفة من الكلمات، وذلك ببرنامج آخر مملّ!

أمّا الآن فيمكنك تقسيم النصّ إلى مجموعة من الكلمات بهذا السطر المنفرد:

**Words() = MyText.Split(Chr(13), " "c)**

طبعا ستحتاج لبذل بعض الجهد الزائد، فالسطر يتميّز بعلامتين: Chr(13) في نهاية السطر، و Chr(10) في بداية السطر التالي، وبعد تنفيذ الجملة السابقة، ستحتوي الكلمات التي كانت في بداية الأسطر ـ من السطر الثاني إلى الأخير ـ على الحرف Chr(10) في بدايتها، وهو طبعا ما لا تريده.. أحد الحلول هو أن تتخلّص من هذا الحرف أولا ثمّ تجري عمليّة التقسيم.. انظر كيف كتبنا كلّ ذلك في سطر واحد:

**Words() = MyText.Replace(Chr(10), "").Split(Chr(13), " "c)**

وطبعا يمكن أن تضع في اعتبارك علامات الترقيم، مثل الأقواس والفاصلة والنقاط وغيرها.

**يبدأ بـ StartsWith وينتهي بـ EndsWith:**

استخدم هاتين الوسيلتين لتتأكّد إذا ما كان النصّ يبدأ بحروف معيّنة أو ينتهي بحروف معيّنة:

**Dim Word As String = "استبدال"**

**Dim Found As Boolean = Word.StartsWith ("اس")**

**Found = Word.EndsWith ("ال")**

وهاتان الدالتان تقومان بعمليّة المقارنة مع مراعاة حالة الأحرف (الإنجليزيّة بالطبع)، ولو لم تكن مهتمًّا بمراعاة الأحرف، فحوّل النصّ أولا إلى الحالة الكبيرة Capital ثمّ استخدم أيا من هاتين الدالتين بعد ذلك.. انظر للمثال التالي:

**Dim Word As String = "test"**

**If Word.ToUpper.StartsWith("TE") Then**

**MsgBox ("نعم يبدأ بهذين الحرفين")**

**End If**

**موضع IndexOf:**

تبحث عن مجموعة من الحروف في النصّ، فإن وجدتها تُرجع موضع أوّل حرف منها في النصّ.

وهي شبيهة بالوسيلة IndexOf الخاصّة بالمصفوفات.

**آخر موضع LastIndexOf:**

مماثلة لتلك الخاصّة بالمصفوفات.. وهي تختلف عن الوسيلة السابقة في أنّها تبدأ البحث من نهاية النصّ إلى بدايته.

ولديك دالة من دوال VB تقوم بنفس هذه الوظيفة، وهي الدالة "في النصّ عكسيّا" InStrRev، وقد تعرفنا عليها من قبل.

**موضع أيّ منها IndexOfAny:**

هذه الوسيلة شيّقة، فهي تمكّنك من البحث عن مجموعة منفصلة من الحروف (نرسلها للدالة في مصفوفة)، بحيث تُعيد هذه الدالة موضع أوّل حرف تجده من هذه الحروف في النصّ.. انظر للمثال التالي:

**Dim chars() As Char = {"."c, " "c, ","c, ";"c}**

**Dim myString As String = "This Is a short sentence"**

**Console.WriteLine(myString.IndexOfAny(chars))**

عند تنفيذ المثال السابق، ستطبع الجملة الأخير الرقم 4 في نافذة المخرجات.. هذا هو موضع المسافة الأولى في النصّ.

**إدراج Insert:**

استخدم هذه الوسيلة لوضع أي مجموعة من الحروف في أي موضع من النصّ، وهي بالصيغة التالية:

**NewString = MyString.Insert(موضع الإدراج, النصّ المدرج)**

**إزالة Remove:**

يمكنك باستخدام هذه الوسيلة إزالة جزء من النصّ، بدءا من موضع معيّن وبطول معيّن:

**newString = str.Remove(موضع بداية الحذف, عدد الحروف)**

**محاذاة يمنى PadRight ومحاذاة يسرى PadLeft:**

يمكنك استخدام الوسيلة PadRight للحصول على نسخة من النصّ وقد أضيف في نهايته عدد من المسافات لإكمال طول النصّ إلى العدد الذي ترغبه من الحروف.. ويحدث المثل عند استخدام الوسيلة PadLeft ولكنّ المسافات تُضاف في بداية النصّ.

وتأخذ هذه الوسيلة معاملا، هو الطول الكلّيّ الذي تريده للنصّ.. انظر للمثال التالي:

**Dim LPString, RPString As String**

**Console.WriteLine("[" & "Mastering VB".PadRight(20) & "]")**

**Console.WriteLine("[" & "Mastering VB".PadLeft(20) & "]")**

بعد تنفيذ هذا الكود ستحصل على النتيجة التالية:

[Mastering VB ]

[ Mastering VB]

حيث تمّ وضع 8 مسافات على يمين النصّ الأوّل، و 8 على يسار الثاني، بحيث صار كلاهما يتكوّن من 20 حرفا.

وتوجد دالتان من دوال VB مناظرتان لهاتين الوسيلتين، هما "ضع على اليمين" RSet و"ضع على اليسار" LSet، تقومان بنفس الوظيفتين.

وتوجد صيغة أخرى لهاتين الوسيلتين، تمكّنك من تحديد الحرف الذي تريد إكمال طول النصّ به:

**LPString = "Mastering VB".PadRight(20, "@")**

**RPString = "Mastering VB".PadLeft(20, ".")**

حيث سيحتوي المتغيران على هذين النصّين:

Mastering VB@@@@@@@@

........Mastering VB

ولكن ماذا سيحدث إذا كان الطول الذي حدّدته أقصر من طول النصّ الأصليّ؟

لن يحدث أيّ شيء.. ستحصل على نفس النصّ الأصليّ!

**الدوال النصّيّة في VB:**

بالإضافة للوسائل التي تمنحها لك خليّة النصوص، يمنحك VB الدوال التالية للتعامل مع النصوص، والتي ستجد أن معظمها يقوم بوظيفة مماثلة لإحدى وظائف وسائل خليّة النصّ.. وعليك أن تنتبه جيّدا إلى أنّ هذه الدوالّ تبدأ ترقيم حروف النصّ بالرقم 1 وليس صفرا:

**LCase:**

تعيد هذه الدالة نسخة من النصّ كلّ حروفها كبير Capital.

**UCase:**

تعيد هذه الدالة نسخة من النصّ كلّ حروفها صغير Small.

**InStr:**

استخدم هذه الدالة للبحث عن نصّ ما داخل نصّ آخر.. وهذه الدالة تأخذ ثلاثة معاملات، هي بالترتيب: رقم الحرف الذي يبدأ عنده البحث داخل النص (وهو معامل اختياريّ، لو لم ترسله فسيتمّ البحث من أوّل حرف)، والنص الذي سيتم البحث فيه، والنص الهدف الذي سيتمّ البحث عنه.. وهي تشبه الوسيلة IndexOf الخاصّة بخليّة النصّ.

وهناك معامل رابع اختياريّ، يسمح لك بتحديد إذا ما كان البحث سيراعي حالة الأحرف CompareMethod.Binary أم لا CompareMethod.Text، وهو ما لا تقدّمه لك الوسيلة IndexOf.

**InStrRev:**

مماثلة للدالة السابقة، إلا إنّها تبدأ البحث من نهاية النصّ لأوّله.

**StrComp:**

مماثلة للوسيلة Compare الخاصّة بخليّة النصّ.

**Left:**

تعيد لك جزءًا من بداية النصّ بالطول الذي تريده.. ونظرا لأنّ هذه الدالة تتشابه في الاسم مع إحدى وسائل النموذج، فلن تستطيع استخدامها في النموذج إلا بالصيغة التالية:

**MsgBox(Microsoft.VisualBasic.Left("محمد", 2)) ' مح**

شيء يدعو للاستياء.. هه؟

حسنا.. خذ هذا الحل: اكتب جملة الاستيراد التالية في بداية الملفّ:

**Imports VB = Microsoft.VisualBasic**

الآن يمكنك استخدام الكود السابق بالصيغة التالية:

**MsgBox(VB.Left("محمد", 2)) ' مح**

**Right:**

تعيد لك جزءًا من نهاية النصّ بالطول الذي تريده.. ونظرا لأنّ هذه الدالة تتشابه في الاسم مع إحدى وسائل النموذج، فلن تستطيع استخدامها في النموذج إلا بالصيغة التالية:

**MsgBox(Microsoft.VisualBasic.Right("محمد", 2)) ' مد**

**Mid:**

تعيد لك جزءًا من أيّ موضع من النصّ بالطول الذي تريده:

**Mid("محمد",2, 3)) ' حمد**

ويمكن استخدام هذه الدالّة في تغيير جزء من النصّ على شرط ألا يتمّ تغيير طول النصّ الأصليّ:

**Mid("محمد",2, 3)) = "اجد" ' سيصبح النصّ هو "ماجد"**

**Len:**

تعيد طول النصّ المرسل لها كمعامل (عدد حروفه).. ويمكن استخدام هذه الدالة بطريقة أخرى، حيث يمكنك إرسال أيّ نوع من المتغيّرات كمعامل لها، لتخبرك بعدد الوحدات Bytes التي يشغلها في الذاكرة.

**LTrim:**

تزيل المسافات الموجودة في بداية النصّ.

**RTrim:**

تزيل المسافات الموجودة في نهاية النصّ.

**Trim:**

تزيل المسافات الموجودة في بداية ونهاية النصّ.

**Space:**

تعيد نصّا عبارة عن مجموعة من المسافات، بالعدد الذي أرسلته للدالة كمعامل:

**Dim X As String = Space(8)**

**StrDup:**

مماثلة للدالة السابقة، ولكنّها تسمح لك بتحديد الحرف الذي سيتمّ تكراره في النصّ بدلا من المسافة:

**MsgBox(StrDup(12, "\*")) ' \*\*\*\*\*\*\*\*\*\*\*\***

**StrConv:**

تقوم هذه الدالة بإجراء عمليّات تحويل متعدّدة على النصّ المرسل لها كمعامل، كما تحدّد أنت في معاملها الثاني:

**StrConv(النصّ, نوع التحويل)**

ويأخذ المعامل الثاني واحدة من قيم المرقّم VbStrConv، وهي كالتالي:

|  |  |
| --- | --- |
| UpperCase | تحويل النصّ لحروف كبيرة. |
| LowerCase | تحويل النصّ لحروف صغيرة. |
| ProperCase | تحويل الحرف الأوّل فقط لحرف كبير. |
| Wide | تحويل الحروف التي تمثّل النصّ من حروف ضيّقة (ممثّلة بوحدة واحدة Byte)، إلى حروف موسّعة (ممثّلة بوحدتين 2 Bytes). |
| Narrow | تحويل الحروف التي تمثّل النصّ من حروف موسّعة، إلى حروف محدودة. |

ويمكن إجراء أكثر من تحويل معا، وذلك بدمج هذه القيم بالمعامل Or أو بعمليّة جمع عاديّة.

**StrReverse:**

تعيد نصّا ترتيب حروفه معاكس لترتيب حروف النصّ المرسل لها كمعامل:

**MsgBox(StrReverse ("تجريب")) ' بيرجت**

**Filter:**

أرسل لهذه الوسيلة مصفوفة نصّيّة، وحدّد القيمة التي تريد البحث عنها في خاناتها، لتحصل على مصفوفة جديدة تحتوي على الخانات التي تحتوي على هذه القيم فقط.. ولهذه الدالة الصيغة التالية:

**Filter(مصفوفة نصّيّة, نصّ البحث[, احتواء][, نوع المقارنة])**

لاحظ أن إرسال المعامل الثالث بالقيمة False يجعل الدالة تعيد الخانات التي لا تحتوي على نصّ البحث.

وطبعا المعامل الرابع يحدّد إذا كان البحث سيراعي حالة الأحرف أم لا.

لاحظ أيضا أنّ هذه الدالة ستبحث عن نصّ البحث كجزء من كلّ خانة.. بمعني أنّك لو بحثت عن النصّ "محمد" وكانت هناك خانة في المصفوفة تحتوي على النصّ "محمد حمدي"، فستعتبر هذه الدالة أنّها عثرت على نصّ البحث في هذه الخانة.

مثال:

**Dim Names() As String = {"علي", "فكري", "زاهر", "سامح", "شريف"}**

**Dim N() As String = Filter(Names, "زاهر")**

**If UBound(N) > -1 Then MsgBox("الاسم موجود")**

**Replace:**

مماثلة لوسيلة خليّة النصّ التي تحمل نفس الاسم، إلا إنّ معاملاتها تختلف بعض الشيء:

**Replace(النصّ الأصليّ, نصّ البحث, استبدال بـ[, موضع بداية البحث][, \_**

**عدد مرّات الاستبدال][, نوع المقارنة])**

**Join:**

مماثلة لوسيلة خليّة النصّ التي تحمل نفس الاسم، إلا إنّ ترتيب المعاملين معكوس:

**Join (مصفوفة نصوص, حروف فاصل)**

**Split:**

مماثلة لوسيلة خليّة النصّ التي تحمل نفس الاسم إلا إنّ معاملاتها تختلف بعض الشيء:

**مصفوفة نصّيّة = Split(النصّ الأصليّ[, حرف فاصل] \_**

**[, عدد خانات المصفوفة الناتجة][, نوع المقارنة])**

**Val:**

تعيد هذه الدالة عددا صحيحا يمثّل العدد الذي يحتويه المتغيّر النصّيّ المرسل لها كمعامل ـ إن كان هذا المتغيّر يحتوي على عدد.. مثال:

**Dim X As String = "5"**

**Dim Y As Integer = Val(X)**

**Str:**

تعيد هذه الدالة التمثيل النصّيّ للعدد المرسل لها كمعامل.. مثال:

**Dim X As Integer = "5"**

**Dim Y As String = Str(X)**

**CStr:**

تقبل هذه الدالة متغيّرا من أيّ نوعٍ، وتعيد لك متغيّرا نصّيّا يمثّل القيمة التي بالمتغيّر الأصليّ ـ إن كانت قابلة للتحويل لنصّ.

**خليّة باني النصّ StringBuilder Class**

* [**وظيفتها**:](file:///C:\Users\Administrator\Desktop\vb.net\c-%20Classes%20الخلايا\08-%20خلايا%20النصوص%20والتواريخ\3-%20خليّة%20باني%20النصّ\1-%20وظيفتها.htm)
* [**الخصائص:**](file:///C:\Users\Administrator\Desktop\vb.net\c-%20Classes%20الخلايا\08-%20خلايا%20النصوص%20والتواريخ\3-%20خليّة%20باني%20النصّ\2-%20الخصائص.htm)
* [**الوسائل:**](file:///C:\Users\Administrator\Desktop\vb.net\c-%20Classes%20الخلايا\08-%20خلايا%20النصوص%20والتواريخ\3-%20خليّة%20باني%20النصّ\3-%20الوسائل.htm)

**وظيفتها:**

عندما تحتاج لبناء نصّ طويل جدا، عن طريق تشبيك أجزائه عبر جملة تكراريّة مثلا، يمكنك استخدام هذه الخليّة فهي أسرع بمراحل من خليّة النصّ العاديّة.

طبعا ستتساءلُ بشكّ:

- ولماذا هذا إن شاء الله؟

- إنّ تساؤلك هذا حقّ لك إن لم يكن واجبا عليك!

- هلاّ دخلتَ في الموضوع؟!

- إنّ ذلك يعود لسمة عجيبة في خليّة النصّ String، فالنصّ الذي تحتوي عليه غير قابل للتغيير!

- نعم؟!!.. أيّ كلامٍ هذا؟.. لطالمنا غيّرنا قيمة النصّ زيادةً ونقصانا دونَ أدنى مشكلة.

- هذا حقيقيّ، ولكنّ المشكلة كلّها تكمن في كيفيّة حدوثه.

- كيف؟

- عندما تحاول تغيير قيمة متغيّر نصّيّ، يتمّ حجز مساحة جديدة في الذاكرة، ونسخ القيمة الجديدة بها، ومحو القيمة القديمة، وربط المتغيّر بالقيمة الجديدة.

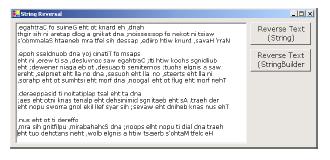
- يا إلهي!.. يا لها من مأساة!

- بالضبط.. هذا هو السبب الذي يجعل تغيير محتوى المتغيّر النصّيّ في جملة تكراريّة طويلةٍ عمليّة انتحاريّة، فهي ترهق VB للغاية كما ترى، وتضيّع الكثيرَ من الوقت.

- وهل يحلّ باني النصّ StringBuilder هذه المشكلة؟

- بالتأكيد.. إنّ هذه الخليّة تمثّل نصّا مرنا قابلا للتغيير دونَ تلك اللّفة الطويلة.

ولكي تدرك فارق السرعة، جرّب مشروع StringReversal في مجلّد برامج هذا الفصل، وهو يتكوّن من مربّع نصّ، وزرين كلاهما يقوم بعكس ترتيب حروف مربع النصّ، ولكنّ أحدهما يقوم بذلك عن طريق متغيّر نصّيّ عاديّ، والآخر يقوم به عن طريق باني النصّ.



ولتجربة البرنامج، حدّد كلمات هذا الفصل وانسخها، ثمّ انتقل إلى المشروع وشغّله، وقف في مربّع النصّ وألصقها (Ctrl+V).. جرّب الزرّ الأوّل، وهو الذي يعكس النصّ باستخدام باني النصّ.. ستجد أنّ النصّ قد تمّ عكسه في أقّل من ثانية.. جرّب الآن الزر الثاني، الذي يقوم بعكس النصّ باستخدام متغيّر نصّيّ.. ستنتظر طويلا جدا لدرجة أنّك ستظنّ أنّ البرنامج قد توقّف عن الاستجابة!.. وحينما تيأس من انتهاء هذه العمليّة، اضغط Ctrl+Break من لوحة المفاتيح لإيقاف التنفيذ.. ستجد أنّ نافذة الكود قد فتحت، وأنّ هناك تظليلا أخضر على أحد سطور جملة التكرار التي تعكس النصّ.. حلّق بالفأرة فوق المتغيّر الذي يمثّل عدّاد التكرار وانتظر لحظةً حتّى تظهر لك قيمته.. في الغالب ستجد أنّه ما زال في بداية الرحلة!!

عامّة لن نطيل في شرح هذه الخليّة، فباني النصّ متشابه كثيرا مع النصّ العاديّ، كما أنّه لا يحتوي على كلّ الوسائل التي تحتويها خليّة النصوص (وسنرى بعد قليلٍ كيف نتعالى على هذا القصور).

في البداية، أضف جملة الاستيراد التالية لبداية الملفّ:

**Imports System.Text**

بعد ذلك يمكنك تعريف متغير من هذه الخليّة بعبارة كالتالية:

**Dim txt As New StringBuilder**

ويمكن إرسال قيمة النصّ لمنشئ الخليّة Constractor عند تعريف المتغيّر كالتالي:

**Dim txt As New StringBuilder("أيّ نصّ")**

في الوضع التلقائيّ، تكون سعة هذا النصّ 16 حرفا، وكلما تطلب الأمر تتمّ مضاعفتها.. ولو أردت أنت تحديد سعة هذا النصّ (أقصى طول له)، فاستخدم خاصيّة "السعة" Capacity.

ولو أردت أن تعرف أقصى سعة متاحة لك، فاستخدم خاصيّة MaxCapacity.

فإذا كان بإمكانك توقّع أطول نصّ ستكتبه في باني النصّ، فيمكنك إرسال هذه السعة لمنشئ الخليّة كالتالي:

**Dim txt As New StringBuilder(السعة الابتدائيّة)**

ولن تكون السعة التي تحدّدها عائقا أمام باني النصّ إذا ما تطلّب الأمر زيادة طول النصّ.. فإذا ما أردت أن تحدد سعة قصوى للنصّ، فاستخدم الصيغة:

**Dim txt As New StringBuilder(السعة الابتدائيّة, السعة القصوى)**

ولو شئت أن تذكر كلّ هذه القيم معا في التعريف فاستخدم الصيغة التالية:

**Dim txt As New StringBuilder(نصّ, السعة الابتدائيّة, السعة القصوى)**

**الخصائص:**

ليس هناك ما نقول عن الخصائص التالية فكلها معروفة:

السعة Capacity ـ السعة القصوى MaxCapacity ـ الطول Length ـ الحروف Chars.

شيء واحد يجب أن تلاحظه، هو أنّ خاصيّة الحروف Chars في باني النصّ تمكّنك من قراءة **وكذلك تغيير** أيّ حرف في النص، بخلاف الوضع في النصّ العاديّ، حيث تقوم هذه الخاصيّة بقراءة الحروف ولكن لا تستطيع تغييرها.. وكمثال، يمكنك تغيير الحرف الثالث في النصّ كالتالي:

**txt.Chars(2) = "H"c**

**الوسائل:**

معظم الوسائل مماثلة لوسائل خليّة النصّ العاديّة، لهذا فسنكتفي بشرح الوسيلتين التاليتين:

**إضافة Append:**

يمكنك استخدام هذه الوسيلة لإضافة أيّ حرف أو نصّ أو رقم أو تاريخ على نهاية النصّ الحاليّ:

**SB.Append(نصّ)**

وهناك صيغة أخرى من هذه الوسيلة، تمكّنك من إضافة مصفوفة من الحروف إلى النصّ كالتالي:

**SB.Append(مصفوفة حروف, خانة البداية, عدد الخانات)**

وهناك صيغة ثالثة تسمح لك بإضافة جزء من نصّ، كالتالي:

**SB.Append(نصّ, موضع البداية, عدد الحروف)**

**حوّل إلى نصّ ToString:**

استخدم هذه الوسيلة للحصول على النصّ الذي تحتويه هذه الخليّة أو أيّ جزء منه، وفي هذه الحالة يمكنك استخدام أيّ وسيلة من وسائل النصّ العاديّ لا يمنحها لك باني النصّ:

**Dim X As New StringBuilder("تجريب")**

**MsgBox (X.ToString.IndexOf("ج").ToString) ' ستعرض الرسالة 1**

**التعامل مع التواريخ**

سنتعرّف هنا على خليتين هامّتين، أولاهما تتعامل مع التواريخ والأوقات وهي الخليّة DateTime، والثانية تستخدم لحساب الزمن المنقضي عن طريق حساب الفرق في الوقت وهي الخليّة TimeSpan.

* [**خليّة التاريخ والوقت DateTime Class:**](file:///C:\Users\Administrator\Desktop\vb.net\c-%20Classes%20الخلايا\08-%20خلايا%20النصوص%20والتواريخ\2-%20التعامل%20مع%20التواريخ\1-%20خليّة%20التاريخ%20والوقت.htm)
* [**خليّة الوقت المنقضي TimeSpan Class:**](file:///C:\Users\Administrator\Desktop\vb.net\c-%20Classes%20الخلايا\08-%20خلايا%20النصوص%20والتواريخ\2-%20التعامل%20مع%20التواريخ\2-%20خليّة%20الوقت%20المنقضي.htm)

**خليّة التاريخ والوقت DateTime**

* [**وظيفتها:**](file:///C:\Users\Administrator\Desktop\vb.net\c-%20Classes%20الخلايا\08-%20خلايا%20النصوص%20والتواريخ\1-%20خليّة%20التاريخ%20والوقت\1-%20وظيفتها.htm)
* [**الخصائص:**](file:///C:\Users\Administrator\Desktop\vb.net\c-%20Classes%20الخلايا\08-%20خلايا%20النصوص%20والتواريخ\1-%20خليّة%20التاريخ%20والوقت\2-%20الخصائص.htm)
* [**الوسائل:**](file:///C:\Users\Administrator\Desktop\vb.net\c-%20Classes%20الخلايا\08-%20خلايا%20النصوص%20والتواريخ\1-%20خليّة%20التاريخ%20والوقت\3-%20الوسائل.htm)

**وظيفتها:**

تعرّفنا على المتغيّرات من النوع Date.. في الواقع لا توجد خليّة بهذا الاسم.. إنّ الخليّة التي يمنحها لك إطار العمل للتعامل مع التواريخ والأوقات هي الخليّة DateTime.. أمّا Date فهو نوع من أنواع البيانات فحسب.. وفي كلتا الحالتين، سواء عرّفت المتغيّر من هذا أم من ذاك، فأنت تستخدم خواصّ ووسائل خليّة التاريخ والوقت، حتّى المشترك منها.. ويمكنك اعتبار الأمر نوعا من اختصار الاسم!

وفي التعريفين التاليين نستخدم الاسمين، حيث لا يوجد أيّ فارق بينهما:

**Dim date1 As DateTime = #4/15/2001#**

**Dim date2 As Date = #4/15/2001 14:01:59#**

**خصائص خليّة التاريخ والوقت:**

يتكون التاريخ من جزأين: تاريخ اليوم Date والوقت Time.. وستجد أنّ مهمّة كلّ الخصائص التالية استخلاص المعلومات المضمّنة في هذا التاريخ.

**التاريخ Date:**

بهذه الخاصيّة يمكنك قراءة الجزء الخاصّ بالتاريخ في المتغيّر، بينما جزء الوقت سيتمّ جعله مساويا لمنتصف الليل (12:00:00 AM).. انظر للمثال التالي:

**Dim date1 As Date**

**date1 = Now()**

**' طبعا سيكون تاريخ اليوم مختلفا عند تنفيذك لهذه الجملة عمّا هو عليه في هذا المثال.**

**Console.WriteLine(date1) ' 5/29/2001 2:30:17 PM الناتج هو**

**Console.WriteLine(date1.Date) ' 5/29/2001 12:00:00 AM الناتج هو**

**الوقت TimeOfDay:**

يمكنك باستخدام هذه الخاصيّة أن تقرأ الجزء الخاصّ بالوقت في التاريخ.

**ترتيب اليوم في الأسبوع DayOfWeek، وترتيب اليوم في السنة DayOfYear:**

ترتيب اليوم في الأسبوع هو رقم من 1 إلى 7، بينما ترتيب اليوم في السنة هو عدد من 1 إلى 365 أو 366 على حسب نوع السنة: كبيسة أم بسيطة.

**ملحوظة:**

يمنحك VB بعض الدوال الجاهزة للتعامل مع التاريخ، مثل "اسم يوم الأسبوع" WeekDayName()، التي ترسل لها رقم اليوم لتذكر لك اسمه.. والدالة "اسم الشهر" MonthName() التي ترسل لها رقم الشهر فتذكر لك اسمه.. وتقبل كلتا الدالتين معاملا اختياريا آخر، إذا جعلت قيمته True ترُجع لك الدالتان اسم الأسبوع أو الشهر مختصرا، وإذا جعلته False تُرجعان لك الاسم كاملا.. كما تقبل دالة "اسم يوم الأسبوع" WeekDayName معاملا ثالثا اختياريا، يمكّنك من تحديد يوم بداية الأسبوع، نظرًا لأنّه في الوضع التلقائيّ يكون يوم الأحد.

**وللحصول على أيّ جزء من أجزاء التاريخ، استخدم الخواصّ التالية:**

العام **Year** ـ الشهر **Month** ـ اليوم **Day** ـ الساعة **Hour** ـ الدقيقة **Minute** ـ الثانية **Second** ـ الجزء من الألف الثانية **Millisecond** ـ الجزء من مئة ألف من الثانية **Ticks**.

**الوسائل**

* [**وسائل التعامل مع التاريخ:**](file:///C:\Users\Administrator\Desktop\vb.net\c-%20Classes%20الخلايا\08-%20خلايا%20النصوص%20والتواريخ\الوسائل\1-%20وسائل%20التعامل%20مع%20التاريخ.htm)
* [**وسائل تحويل التاريخ:**](file:///C:\Users\Administrator\Desktop\vb.net\c-%20Classes%20الخلايا\08-%20خلايا%20النصوص%20والتواريخ\الوسائل\2-%20وسائل%20تحويل%20التاريخ.htm)

**وسائل التعامل مع التاريخ:**

**عدد أيّام الشهر DaysInMonth:**

تخبرك هذه الدالة بعدد أيّام الشهر الذي تريده.. ونظرا لأنّ شهر فبراير يكون 28 يوما في السنة البسيطة و29 يوما في السنة الكبيسة، فإنّ من الطبيعيّ أنّ تحتاج هذه الدالة لإرسال السنة إليها كمعامل، بجانب إرسال اسم الشهر:

**monDays = DateTime.DaysInMonth(السنة, رقم الشهر)**

**إنّها سنة كبيسة IsLeapYear:**

إذا كانت السنة تقبل القسمة على 4، فإنّ ذلك يعنى أنّها سنة كبيسة، بمعنى أنّ عدد أيّامها هو 366، حيث يكون شهر فبراير في هذه السنة 29 يوما.. إنّ السبب في هذه القفزة في التقويم يرجع إلى أنّ السنة الحقيقيّة (مدّة دوران الأرض حول الشمس) هي 365.25 يوما.. لهذا كان الحلّ هو إضافة يوم كل أربع سنوات لتصحيح التقويم.

استخدم هذه الوسيلة لتعرف نوع السنة.

**إضافة Add:**

هذه الوسيلة تمكّنك من تكوين تاريخ جديد، ناتج عن إضافة المدّة الزمنيّة التي يحتويها كائن الوقت "المنقضي" TimeSpan، للنسخة الحاليّة من التاريخ.. وسنتعرّف علي الخليّة TimeSpan لاحقا في هذا الفصل.

**طرح Subtract:**

هذه هي الوسيلة العكسيّة للوسيلة السابقة، فهي تمكّنك من طرح تاريخين، حيث يعود لك الفارق بينهما في صورة كائن من النوع TimeSpan.

**ولإضافة أجزاء زمنيّة إلى التاريخ، يمكنك أن تستخدم هذه الوسائل:**

إضافة سنين **AddYears** ـ إضافة شهور **AddMonths** ـ إضافة أيّام **AddDays** ـ إضافة ساعات **AddHours** ـ إضافة دقائق **AddMinutes** ـ إضافة ثوانيَ **AddSeconds** ـ إضافة أجزاء من الألف من الثانية **AddMilliseconds** ـ إضافة أجزاء من مئة ألف من الثانية **AddTicks**.

ولست في حاجة لإخبارك أنّك تستطيع جعل معامل أيّ من هذه الدوال سالبا، بحيث تقوم بطرح المدّة المحدّدة من التاريخ بدلا من إضافتها.

**وسائل تحويل التاريخ:**

**حوّل إلى نصّ ToString:**

بالطبع صارت هذه الوسيلة لديك أشهر من راقصة في مصر! (أو من نار على علم لمن لا يحبّون تحديث الأمثال!).. وما يهمّ هنا، هو أن أذكّرك أن هذه الوسيلة يمكن أن تستقبل منك معاملا يحدّد التنسيق الذي سيتمّ به تحويل التاريخ إلى نصّ.. ولمزيد من التفاصيل حول كيفية هذا التنسيق، يمكنك الرجوع لملفات إرشادات اللغة.

وسأكتفي هنا بعرض الأمثلة التالية، متأكّدا أنّك ممّن يحبّون المثل القائل: "يدرك الذكيّ بالمثال الواحد، ما لا يدركه الغبيّ بألف مثال"!.. وستجد ناتج كلّ جملة مكتوبا بعدها كتعليق:

**Console.WriteLine(now().ToString("d"))**

**' 5/29/2001**

**Console.WriteLine(now().ToString("D"))**

**' Tuesday, May 29, 2001**

**Console.WriteLine(now().ToString("f"))**

**' Tuesday, May 29, 2001 3:14 PM**

**Console.WriteLine(now().ToString("F"))**

**' Tuesday, May 29, 2001 3:14:43 PM**

**Console.WriteLine(now().ToString("g"))**

**' 5/29/2001 3:14 PM**

**Console.WriteLine(now().ToString("G"))**

**' 5/29/2001 3:14:43 PM**

**Console.WriteLine(now().ToString("m"))**

**' May 29**

**Console.WriteLine(now().ToString("r"))**

**' Tue, 29 May 2001 15:14:43 GMT**

**Console.WriteLine(now().ToString("s"))**

**' 2001-05-29T15:14:43**

**Console.WriteLine(now().ToString("t"))**

**' 3:14 PM**

**Console.WriteLine(now().ToString("T"))**

**' 3:14:43 PM**

**Console.WriteLine(now().ToString("u"))**

**' 2001-05-29 15:14:43Z**

**Console.WriteLine(now().ToString("U"))**

**' Tuesday, May 29, 2001 12:14:43 PM**

**Console.WriteLine(now().ToString("y"))**

**' May, 2001**

**Console.WriteLine(now().ToString("m/d/yyyy"))**

**' 5/29/2001**

**Console.WriteLine(now().ToString("dd"))**

**' 29**

**Console.WriteLine(now().ToString("ddd"))**

**' Tue**

**Console.WriteLine(now().ToString("dddd"))**

**' Tuesday**

**Console.WriteLine(now().ToString("MMMM d"))**

**' May 29**

**Console.WriteLine(now().ToString("M/yyyy"))**

**' May 2001**

**Console.WriteLine(now().ToString("MM"))**

**' 05**

**Console.WriteLine(now().ToString("MMM"))**

**' May**

**Console.WriteLine(now().ToString("MMMM"))**

**' May**

**Console.WriteLine(now().ToString("m/d/y"))**

**' 29/5/1**

**Console.WriteLine(now().ToString("m/d/yy"))**

**' 29/5/01**

**Console.WriteLine(now().ToString("yy"))**

**' 01**

**Console.WriteLine(now().ToString("yyyy"))**

**' 2001**

**Console.WriteLine(now().ToString("gg"))**

**' A.D.**

**Console.WriteLine(now().ToString("hh"))**

**' 03**

**Console.WriteLine(now().ToString("HH"))**

**' 15**

**Console.WriteLine(now().ToString("h:m"))**

**' 3:26**

**Console.WriteLine(now().ToString("mm"))**

**' 26**

**Console.WriteLine(now().ToString("h:m:s"))**

**' 3:6:8**

**Console.WriteLine(now().ToString("hh:mm:ss"))**

**' 03:06:08**

**Console.WriteLine(now().ToString("hh:mm:ss.fff"))**

**' 03:06:08.122**

**Console.WriteLine(now().ToString("h:m:s t"))**

**' 3:26:38 P**

**Console.WriteLine(now().ToString("tt"))**

**' PM**

**Console.WriteLine(now().ToString("zz"))**

**' +03**

**Console.WriteLine(now().ToString("zzz"))**

**' +03:00**

وليس التحويل إلى نصّ هو كل ما تستطيعه، فلديك أيضا هذه الوسائل، للتحويل إلى الأنواع الرقميّة:

**ToInt16** ـ **ToSingle** ـ **ToUInt16** ـ **ToUInt32** ـ **ToUInt64**.

مع ملاحظة أنّ التقويم يُمثّل بعدد مزدوج، الجزء الصحيح به هو التاريخ، والجزء العشريّ هو الوقت.. لهذا فمن الطبيعيّ عند تحويل التقويم إلى عدد صحيح أن يضيع منه الوقت.

وهناك تحويلات أخرى مثل:

**التحويل بين التاريخ وتعبير OLE:**

نظرا لأنّ VS تقدّم طريقة جديدة للتعامل مع التواريخ، فلا بدّ من أن تمنحك وسيلة للتنسيق مع البرامج الأخرى التي تتعامل مع التاريخ بطريقة مختلفة.. لهذا يمكنك استخدام الدالة **FromOADate** لتحويل التاريخ المستخدم في تقنية OLE والذي هو عبارة عن رقم مزدوج Double، إلى التاريخ المستخدم في خليتنا هذه.. كما يمكنك استخدام الدالة **ToOADate** للقيام بالعكس.

**التحويل بين التاريخ وتنسيق ملفّ الوقت المستخدم في نظامك:**

لديك هنا الوسيلتان **ToFileTime** و **FromFileTime**.

**التحويل بين التوقيت المحلّيّ والعالميّ:**

لديك هنا الوسيلتان **ToLocalTime** و **ToUniversalTime**.

**التحويل إلى نصّ تاريخٍ طويل أو قصير:**

لديك هنا الوسيلتان **ToLongDateString** و **ToShortDateString**.. انظر للمثال التالي:

**Console.WriteLine(Now().ToLongDateString)**

**' Friday, July 16, 2001**

**Console.WriteLine(Now().ToShortDateString)**

**' 7/16/2001**

**التحويل إلى نصّ وقتٍ طويل أو قصير:**

لديك هنا الوسلتان: **ToLongTimeString** و **ToShortTimeString**.. انظر للمثال التالي:

**Console.WriteLine(Now().ToLongTimeString)**

**' 6:40:53 PM**

**Console.WriteLine(Now().ToShortTimeString)**

**' 6:40 PM**

ولمزيد من التدريب، يمكنك اختبار التطبيق StopWatch في مجلد هذا الفصل.

**الدوالّ الزمنيّة:**

بالإضافة لخصائص ووسائل خليّة التاريخ والوقت، فإنّ VB يمنحُك مجموعة أخرى من الدوالّ، منها ما يمكنك استنتاج وظيفته بمجرّد النظر، مثل الدوالّ التالية:

الآن **Now** ـ اليوم **Day** ـ يوم الأسبوع **Weekday** ـ الشهر **Month** ـ اسم الشهر **MonthName** ـ العام **Year** ـ الساعة **Hour** ـ الدقيقة **Minute** ـ الثانية **Second**.

ومنها ما سنوضحه لك ها هنا:

**تكوين التاريخ DateSerial:**

تستقبل هذه الدالة السنة والشهر واليوم، وتعيد لك التاريخ الذي يتكوّن منها.. مثال:

**MsgBox(DateSerial(2002, 10, 1)) '10/1/02**

ولو زاد معامل اليوم عن أقصى يوم في الشهر، أو زاد معامل الشهر عن 12، فإنّ هذه الدالة تقوم بضبط التاريخ، بزيادة الشهر أو السنة.. مثال:

**MsgBox (DateSerial(1996, 1, 90) '3/30/1996**

**قيمة التاريخ DateValue:**

تستقبل هذه الدالة نصّا، وتعيد التاريخ الذي يمثّله هذا النصّ.

**تكوين الوقت TimeSerial:**

تستقبل هذه الدالة ثلاثة معاملات: الساعة والدقيقة والثانية، وتعيد الوقت الذي يتكوّن منها.

**قيمة الوقت TimeValue:**

تستقبل هذه الدالة نصّا، وتعيد الوقت الذي يمثّله هذا النصّ.

**جمع تاريخ DateAdd:**

استخدم هذه الدالة لجمع مدّة معيّنة على أحد التواريخ، تبعا للصيغة التالية:

**التاريخ الجديد = DateAdd (المدى الزمنيّ, القيمة المضافة, التاريخ)**

حيث المدى الزمنيّ هو أحد قيم المرقّم DateInterval التالية:

|  |  |
| --- | --- |
| Year | عام. |
| Quarter | فصل (ربع عام: 1 ـ 4 شهور). |
| Month | شهر. |
| DayOfYear | يوم من السنة (1 ـ 266 يوما). |
| Day | يوم من الشهر (1 ـ 31 يوما). |
| WeekDay | يوم من الأسبوع (1 ـ 7 أيام). |
| WeekOfYear | أسبوع من العام (1 ـ 53 أسبوعا). |
| Hour | ساعة. |
| Minute | دقيقة. |
| Second | ثانية. |

**الفارق الزمنيّ DateDiff:**

تعيد هذه الدالة الفارق الزمنيّ بين تاريخين، تبعا للصيغة التالية:

**الفارق الزمنيّ = DateDiff(المدى الزمنيّ, التاريخ الأوّل, التاريخ الثاني \_**

**[, أوّل يوم في الأسبوع[, أوّل أسبوع في السنة]])**

مثال:

**MsgBox(DateDiff(DateInterval.Day, #1/1/2000#, Now()))**

**جزء من التاريخ DatePart:**

لهذه الدالة الصيغة التالية:

**DatePart(الجزء المطلوب, التاريخ[,أوّل أيّام الأسبوع[, أوّل أسابيع السنة]])**

وهي تعيد جزءا من التاريخ، على حسب قيمة المعامل الأوّل، وهو بالمناسبة معامل نصّيّ.. انظر للأمثلة التالية:

**Console.WriteLine(DatePart("yyyy", Now()))**

**'2001 العام**

**Console.WriteLine(DatePart("q", Now()))**

**'3 رقم الفصل في العام**

**Console.WriteLine(DatePart("m", Now()))**

**'10 رقم الشهر في العام**

**Console.WriteLine(DatePart("y", Now()))**

**'296 رقم اليوم في العام**

**Console.WriteLine(DatePart("d", Now()))**

**'23 رقم اليوم في الشهر**

**Console.WriteLine(DatePart("w", Now()))**

**'3 رقم اليوم في الأسبوع**

**Console.WriteLine(DatePart("ww", Now()))**

**'43 رقم الأسبوع في السنة**

**Console.WriteLine(DatePart("h", Now()))**

**'15 الساعة**

**Console.WriteLine(DatePart("n", Now()))**

**'3 الدقيقة**

**Console.WriteLine(DatePart("s", Now()))**

**'30 الثانية**

وإذا لم تكن مقتنعا باستخدام هذه الرموز، فيمكنك استخدام الصيغة الثانية من هذه الدالة، حيث إنّ معاملها الأوّل من نوع المرقّم DateInterval الذي شرحناه سابقا.

**خليّة الوقت المنقضي TimeSpan Class**

* [**وظيفتها:**](file:///C:\Users\Administrator\Desktop\vb.net\c-%20Classes%20الخلايا\08-%20خلايا%20النصوص%20والتواريخ\2-%20خليّة%20الوقت%20المنقضي\1-%20وظيفتها.htm)
* [**الخصائصّ:**](file:///C:\Users\Administrator\Desktop\vb.net\c-%20Classes%20الخلايا\08-%20خلايا%20النصوص%20والتواريخ\2-%20خليّة%20الوقت%20المنقضي\2-%20الخصائص.htm)
* [**الوسائل:**](file:///C:\Users\Administrator\Desktop\vb.net\c-%20Classes%20الخلايا\08-%20خلايا%20النصوص%20والتواريخ\2-%20خليّة%20الوقت%20المنقضي\3-%20الوسائل.htm)

**وظيفتها**:

يمكنك استخدام هذه الخليّة لحساب مدّة زمنيّة معيّنة، بأيّ وحدة زمنيّة تريدها.

في البداية عرّف متغيرا من هذه الخليّة كالتالي:

**Dim TS As New TimeSpan**

ويمكنك أن تمنح المتغيّر قيمة ابتدائيّة أثناء تعريفه.. والجملة التالية تضع في المتغيّر 9 أيام و 12 ساعة و دقيقة و 59 ثانية:

**Dim TS As TimeSpan = New TimeSpan(9, 12, 1, 59)**

ولقد رأينا الوسيلة Date.Subtract التي تُعيد الفرق بين تاريخين في صورة كائن من هذا النوع.. انظر للمثال التالي:

**Dim TS As New TimeSpan**

**Dim date1 As Date = #4/11/1985#**

**Dim date2 As Date = Now()**

**TS = date2.Subtract(date1)**

**Console.WriteLine(TS)**

طبعا سيتوقّف الناتج على تاريخ اليوم الذي تنفّذ فيه هذا المثال، ولكنّ النتيجة على أيّ حال ستكون على الصورة التالية:

5992.15:58:14.4766848

حيث يمثّل الرقم 5992 عدد الأيّام، وتلاحظ أنّه مفصول عمّا يليه بنقطة "."، بينما باقي الأرقام تدلّ على الوقت، وهي مفصولة عن بعضها بنقطتين متعامدتين ":"، فلدينا هنا مثلا 15 ساعة، و 58 دقيقة و 14.4766848 ثانية.

**الخصائص Properties:**

**TicksPerDay:**

تعطيك الرقم الناتج عن تقسيم اليوم إلى أجزاء من مئة ألف من الثانية!

**TicksPerHour:**

تعطيك الرقم الناتج عن تقسيم الساعة إلى أجزاء من مئة ألف من الثانية!

**TicksPerMinute:**

تعطيك الرقم الناتج عن تقسيم الدقيقة إلى أجزاء من مئة ألف من الثانية!

**TicksPerSecond:**

تعطيك الرقم الناتج عن تقسيم الثانية إلى أجزاء من مئة ألف من الثانية.. وهو طبعا مئة ألف!

**TicksPerMillisecond:**

تُرجع لك 100.

**Zero:**

مدةّ زمنيّة تساوي صفرا.

**المدّة Duration:**

لقراءة المدّة الزمنيّة التي يحتويها هذا الكائن.

**وللحصول على أيّ جزء من أجزاء المدّة الزمنيّة، استخدم الخواصّ التالية:**

الأيّام **Days** ـ الساعات **Hours** ـ الدقائق **Minutes** ـ الثواني **Seconds** ـ الأجزاء من الألف من الثانية **Milliseconds** ـ الأجزاء من المئة ألف من الثانية **Ticks**.

**ولحساب المدّة الزمنيّة بوحدة معيّنة، استخدم هذه الخواصّ:**

المدّة بالأيّام **TotalDays** ـ المدّة بالساعات **TotalHours** ـ المدّة بالدقائق **TotalMinutes** ـ المدّة بالثواني **TotalSeconds** ـ المدّة بالأجزاء من الألف من الثانية **TotalMilliseconds**.

والمثال التالي يريك كيف تستخدم هذه الخليّة لحساب الوقت المنقضي:

**Dim T1, T2 As Date**

**T1 = Now ' وقت البداية**

**MsgBox("انتظر لفترة، ثمّ اضغط موافق لمعرفة الوقت الذي انتظرته")**

**T2 = Now**

**Dim TS As TimeSpan = T2.Subtract(T1)**

**Console.WriteLine("المدّة الكلّيّة = " & TS.Duration.ToString)**

**Console.WriteLine("الدقائق = " & TS.Minutes.ToString)**

**Console.WriteLine("الثواني = " & TS.Seconds.ToString)**

**Console.WriteLine("الأجزاء من الألف من الثانية = " & \_**

**TS.TotalMilliseconds.ToString)**

**Console.WriteLine("الأجزاء من مئة ألف من الثانية = " & TS.Ticks.ToString)**

**Console.WriteLine("المدّة الزمنيّة بالثواني = " & TS.TotalSeconds.ToString)**

**الوسائل:**

**تحويل نصّيّ Parse:**

استخدم هذه الوسيلة لتحويل نصّ يمثّل مدّة زمنيّة، إلى مدّة زمنيّة.. مثال:

**Dim SP As New TimeSpan()**

**SP = TimeSpan.Parse("3.12:20:30.500")**

**Console.WriteLine(SP)**

**' 3.12:20:30.5000000**

**إضافة Add:**

لإنتاج مدّة زمنيّة جديدة، عبارة عن ناتج جمع مدّة زمنيّة معيّنة على النسخة الحاليّة من هذه الخليّة.. مثال:

**Dim TS1 As New TimeSpan("1:00:01")**

**Dim TS2 As New TimeSpan("2:01:09")**

**Dim TS As New TimeSpan = TS1.Add(TS2)**

**طرح Subtract:**

عكس العمليّة السابقة، حيث تُرجع مدّة زمنيّة جديدة، ناتجة عن طرح مدّة زمنيّة معيّنة من المدّة الزمنيّة الحالية.. مثال:

**Dim T1, T2 As TimeSpan**

**T1 = New TimeSpan(3, 9, 10, 12)**

**T2 = New TimeSpan(0, 1, 0, 59, 3)**

**Dim TS As TimeSpan = T2.Subtract(T1)**

**ويمكنك استخدام الوسائل التالية لتكوين مدّة زمنيّة محدّدة:**

بالأيّام **FromDays**، بالساعات **FromHours**، بالدقائق **FromMinutes**، بالثواني **FromSeconds**، بالأجزاء من الألف من الثانية **FromMilliseconds**، بالأجزاء من المئة ألف من الثانية **FromTicks**.. مثال:

**Dim TS As New System.TimeSpan()**

**TS = TimeSpan.FromHours(3)**

**TS = TS.Add(TimeSpan.FromMinutes(2))**

**TS = TS.Add(TimeSpan.FromSeconds(16))**

**TS = TS.Add(TimeSpan.FromMilliseconds(500))**

**Console.WriteLine("المدّة الزمنيّة الكلّيّة هي " & TS.ToString)**

**' 03:02:16.5000000.**

**المدة السالبة Negate:**

استخدم هذه الوسيلة للحصول على القيمة السالبة من المدّة الزمنيّة، وبهذا لو جمعتها على التاريخ الحاليّ فكأنّك تقوم بعمليّة طرح.

ولمزيد من التدريب على استخدام هذه الخليّة في حساب الوقت الذي تستغرقه عمليّة ما، تفحّص مشروع StringReversal في مجلّد هذا الفصل، حيث استخدمنا هذه الخليّة في حساب الوقت اللازم لعكس النصّ بكلتا الخليّتين: String و StringBuilder.