

الفصل الثامن التعامل مع النصوص والخطوط والألوان

في هذا الفصل نتعرض بالتفصيل للتعامل مع النصوص، بدءاً من تمثيلها الداخلي بالاقتراس والتعديل والبحث، حتى مظهرها الخارجي بالخط وحجمه وما إلى ذلك. أيضاً نتعرض بشئ من التفصيل لأداة النص المنوع ثم نعرض كيفية تغيير ألوان العناصر سواءً أثناء التصميم أو أثناء التشغيل. بالانتهاء من هذا الفصل ستعرف على:

- ◆ العمل مع النصوص بواسطة عناصر التحكم.
- ◆ السيطرة على النصوص في **Visual Basic**
- ◆ التحكم في مظهر النص باستخدام الخطوط.
- ◆ إعطاء المستخدم إمكانية التحكم في خط النص.
- ◆ استخدام الألوان لتحسين مظهر واجهة البرنامج.

عندما نذكر كلمة "نص" فإننا لا نعني فقط النص المعروض علي الشاشة في أي عنصر تحكم، ولكننا نعني أيضاً النصوص المخزونة والموجودة في البرنامج. والنص هو وسيلة الاتصال الأولى لغالبية البرامج ولذا يحتل من الأهمية موقعا كبيرا.

ولكي تتمكن من صياغة أي برنامج، ينبغي أن تعرف جيداً كيف تتعامل مع النصوص.

وقد تعاملنا في الفصول السابقة مع العناوين **Labels** ومربعات النص **Text Boxes** وسنتوسع في هذا الفصل في دراسة كيفية تأثير الخواص والدوال على النصوص وكيفية التحكم في الخط ولون النصوص المعروضة في البرنامج.

إدخال النص وعرضه

إن من أبسط المتطلبات في البرنامج إمكانية التخزين ثم عرض المعلومات، وهي غالباً ما تكون في هيئة نصوص، وتتراوح سهولة وبساطة العملية بين عرض اسم في قائمة مثلاً، إلى مقارنة الكلمة والبحث عن إحدى مترادفاتهما. ولكي يمكن إدخال معلومة ما إلى الحاسب، توجد طرق عديدة لعل أبسطها استخدام "مربع النص" **Text Box**. وستعرض لمجموعة أخرى من عناصر التحكم التي تستخدم في ذلك.

أما بالنسبة لعرض المعلومة فيمكن أن نعرضها على صورة عنوان **"Label"** أو داخل مربع النص **"Text Box"**، أو حتى بكتابتها مباشرةً على شريط عنوان النافذة.

وتتميز أداة العنوان **"Label"** بأنها تعرض النص سواء المحفوظ فيها من وقت التصميم أو الذي قام البرنامج أثناء تشغيله بحفظه فيه باستخدام الخاصية **"Text"** ويمكن للعنصر **"Label"** عرض النص في سطر واحد أو عدة أسطر لكنه لا يسمح بالتحرك بأشرطة تمرير كما في مربع النص.

وعلى العكس فإن مربع النص يمكنه فعل كل ما تؤديه أداة العنوان **"Label"** بالإضافة إلى تمكين المستخدم من إدخال نص أثناء التشغيل واستخدام شريط التمرير **Scroll Bar**. تستخدم الخاصية **"Text"** في قراءة المدخلات النصية من مربع النص، كما تستخدم في عرض نص ما في مربع النص.

استخدام خصائص أخرى لمربع النص Text Box

هناك خصائص أخرى لمربع النص وهي كما يلي:

- الخاصية **ReadOnly** : وتمنع المستخدم من إدخال أي نص.
- الخاصية **Maxlength**: وتحدد الحد الأقصى لعدد الأحرف التي يمكن إدخالها لمربع النص.
- الخاصية **Passwordchar** : وهي تخفي الأحرف التي يدخلها المستخدم (تعرض بدلا منها علامة تقوم أنت بتحديددها مثل الحرف *).
- الخاصية **SelectionLength**: وتمكن المبرمج من تحديد عدد الأحرف التي تم اختيارها (تعليمها) من مربع النص.
- الخاصية **SelectionStart** : وهي تحدد الحرف الذي بدأ الاختيار عنده في مربع النص.
- الخاصية **SelectedText** : وهي التي تتعامل مع الجزء المختار من داخل مربع النص.

منع التعديل في مربع النص بالخاصية **ReadOnly** :

عندما تحتاج لعرض نص كبير الحجم_مثلا معلومات عن البرنامج_ لن يناسبك أداة العنوان **Label** حيث أن المعلومات قد تتجاوز حدود النافذة. يمكنك عندئذ استخدام مربع النص، ويسمح مربع النص حينئذ بعرض النص في المساحة المتاحة، ويمكن استخدام أشرطة التمرير لعرض الجزء الغير ظاهر من النص، وتستخدم الخاصية **ReadOnly** لمنع المستخدم من التعديل في النص أو مسحه. بالرغم من ذلك فانه مازال يمكنه التعليم من مربع النص ونسخ النص إلى الحافظة.

يجب عدم الخلط بين الخاصية **ReadOnly** والخاصية **Enabled**. فكل من الخاصيتين يتسبب في تعميم النص خلفية مربع النص. ولكن في حالة **ReadOnly = True** لا يتم تعميم النص نفسه ولكن فقط يمنع المستخدم من تغيير أي نص في هذا المربع. أما في حالة **Enabled = False** فإن مربع النص لا يمكن التعامل معه بأي شكل من الأشكال، ولا يمكن حتى نسخ أي نص منه إلى الحافظة.



مثال : انظر الكود الآتي:

TextBox1.ReadOnly=TURE

TextBox1.Text= " لن تستطيع تعديل هذا النص "

أحد الاستخدامات الجيدة للخاصية **ReadOnly** هو منع المستخدم من التغيير الغير مقصود في محتويات مربعات إدخال في طور العرض. مثال لذلك نافذة إدخال بيانات في طور العرض، يمكن إغلاق إمكانية التعديل ثم إذا أراد المستخدم التعديل يضغط على زر فيتم تغيير الخاصية **ReadOnly** لمربعات الإدخال إلى **False** كما في شكل ٨-١.



شكل ٨-١ شاشة إدخال بيانات تحتوي على الزر "تعديل" الذي يسمح بتعديل المدخلات

تحديد عدد الأحرف في مربع النص

في كثير من الأحيان تحتاج إلى تحديد طول معين للمدخلات. مثلاً عند إدخال كود منطقة في دليل التليفونات فإننا نحدد خانيتين فقط أو ثلاثة، أو عند كتابة رقم الصنف مثلاً. وتزداد أهمية تحديد طول معين للنص في حالة استخدام قواعد البيانات حيث يكون كل حقل **Field** محدد الطول لحساسية هذه الملفات لطول النص إذ يؤثر ذلك في الحجم الكلي للملف.

لتنفيذ ذلك فإننا نستخدم الخاصية **MaxLength** حيث يسمح مربع النص المضبوط على **MaxLength = 7** على إدخال حتى ٧ أحرف فقط وعند تجاوز المستخدم هذا العدد من الأرقام فإن مربع النص لا يستجيب ويتوقف عن إضافة الأحرف مصدرا صوت صغير.

بالرغم من أن مربع النص يمكنه أن يعرض أسطر عديدة إلا أنه في الحقيقة لا يمكنه التعامل مع أكثر من ٣٢.٧٦٧ حرف وتكون في حالة ضبط الخاصية **Multiline = True**. أما في حالة **Multiline = False** فتكون إمكانية مربع النص محدودة بحسب الذاكرة المتاحة في النظام.



إخفاء محتويات مربع النص

في كثير من الأحيان وخاصة في الشبكات يحتاج الولوج إلى الشبكة إلى إدخال كلمة سر وفي هذه الحالة يجب إخفاء الأحرف التي أدخلها المستخدم عن الأعين ونستعوض عنها بالرمز "*" ولجعل مدخلات مربع النص في هذه الصورة فإننا نستخدم الخاصية **PasswordChar**. ونضبطها على الحرف المراد إظهاره بدلا من الحروف المدخلة مثل * مثلا (انظر شكل ٨-٢).



شكل ٨-٢ أحد مربعات النصوص التي تستخدم لإدخال كلمة سر

تعديل محتويات مربع النص

إذا كنت قد تعاملت مع أي معالج نصوص في **Windows** فإنك بالتأكيد تعلم كيف يمكنك نسخ نص واختياره وقصه وما إلى ذلك. وبنفس الطريقة يمكن اختيار النصوص داخل مربع النص. فاختيار جزء معين يتم بواسطة الفأرة أو باستخدام مفتاح **Shift** ومفاتيح الأسهم بينما يكون النسخ إلى الحافظة بواسطة الاختصار **Ctrl + C** أما القص إلى الحافظة فيكون بواسطة الاختصار **Ctrl + X** بينما يكون اللصق من الحافظة بواسطة

الاختصار **Ctrl+V**. والتراجع عن آخر عملية يكون بواسطة الاختصار **Ctrl+Z**.
ويسمح **Windows** بإظهار قائمة موضعية عند الضغط على الزر الأيمن للفأرة داخل مربع النص. حيث تحتوي هذه القائمة على جميع الأوامر السابقة (انظر شكل ٨-٣).



شكل ٨-٣ قائمة التعديل الموضعية التي تظهر بنقر مربع النص بالزر الأيمن للفأرة
كل ما سبق ذكره يعينك كمستخدم للبرنامج الذي يحتوي على مربع النص، وأما في حالة البرمجة فإننا نهتم بنقاط أخرى نوضحها فيما يلي:
معرفة بداية الجزء الذي تم تحديده من قبل المستخدم: ويتم بالخاصية **SelectionStart** التي تعطي رقم يعبر عن ترتيب الحرف الذي بدأ عنده الاختيار (يكون أول حرف في مربع النص بالقيمة صفر).
تحديد طول النص الذي تم اختياره: ويتم ذلك بالخاصية **SelectionLength** التي تعطي رقم يعبر عن عدد الأحرف التي تم اختيارها.
تحديد النص نفسه الذي تم اختياره: وذلك بالخاصية **SelectedText** وهي تعطي الجزء المحدد فقط.

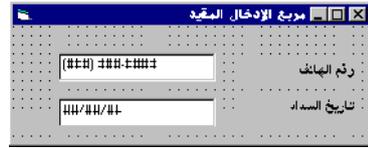
ويمكن استخدام الخواص السابقة بالعكس أي تحديدها بدلاً من قراءتها أثناء تشغيل البرنامج. بمعنى جعل الكود يختار جزء من النص، ومثال ذلك ما يحدث إذا كنت تريد المستخدم أن يكتب فوق محتوى مربع النص بمجرد نقل التركيز إليه بحيث يتم تحديد النص داخل المربع بالكامل مما يجعل ضغط أي حرف من المستخدم يزيل محتوى مربع النص

لإظهار النص الجديد. وعلى الرغم من أن هذه العملية أصبحت من السمات المبنية داخل جميع مربعات النصوص بدءاً من **Visual Basic.Net**، فكلما قمت بنقل التركيز على مربع النص يتم اختيار نص المربع بالكامل إلا أننا سنوضح الكود اللازم لأداء هذه العملية باستخدام الخصائص **SelectionStart** و **SelectionLength** كما يلي:

```
Private Sub TextBox1_Enter(ByVal sender As Object, ByVal e As System.EventArgs) Handles TextBox1.Enter
    TextBox1.SelectionStart = 0
    TextBox1.SelectionLength = Len(TextBox1.Text)
End Sub
```

استخدام حروف نائية لتقييد النص

يستخدم مربع الإدخال أحياناً لتحديد شكل معين للمدخلات أو للنص المدخل، فمثلاً عند إدخال تاريخ يفضل ألا يسمح إلا بشكل معين للمدخلات، ويسهل ذلك كثيراً من عملية التحقق من المدخلات، حيث يتم اختبار القيم وليس نوعية المدخلات، فلن يستطيع المستخدم إلا إدخال تاريخ. إذا حددت له ذلك يسمى مربع النص "مربع الإدخال المقيد" أو **Masked TextBox**. يوضح شكل ٨-٤ كيف تقييد مربع الإدخال ليقبل رقم الهاتف بشكل معين (ثلاثة أرقام ثم سبعة) وكذلك التاريخ (رقمين لليوم ورقمين للشهر ورقمين للسنة مفصولين بعلامة أشرطة).



شكل ٨-٤ مربع الإدخال المقيد يستخدم لتحديد شكل معين للمدخلات.

ولإضافة هذا العنصر إلى المشروع يتم اختياره من مربع الأدوات ثم إدراجه إلى النموذج والتعامل معه كمربع النص تماماً.

لا يستطيع مربع الإدخال المقيد التعامل مع عدد أحرف أكثر من ٦٤ وللتعامل مع نصوص أكبر نرجع إلى استخدام مربع النص العادي والتحكم فيه عن طريق الكود.



يتيح لك مربع الإدخال المقيد تحديد الشكل الذي تريده للمدخلات من خلال خاصية تسمى القناع Mask وهو عبارة عن أحرف مكتوبة بطريقة معينة تمثل نموذج للنص المسموح بإدخاله. على سبيل المثال القيمة ##### للقناع تتيح إدخال عدد مكون من ثلاثة أرقام صحيحة وثلاثة أرقام عشرية بينهم نقطة عشرية. أي أن الرمز # يعني السماح بإدخال أي رقم من ١ إلى ٩ مكانه.

لو حاول المستخدم تجاوز القيم المسموح بها للحرف سيصدر المربع صوت صفير للتنبيه وسيهمل الحرف الخاطئ (يعتمد ذلك حقيقةً على قيمة الخاصية BeepOnError المصاحبة للأداة). بالطبع تحديد قيمة القناع أسهل بكثير من إجراء اختبار على القيم المدخلة بحيث تعطي نفس الناتج.

وإنشاء القناع سهل للغاية، فكل حرف فيه يعني السماح لمجموعة معينة من الأحرف بأن تحل محله في الخاصية Text. أهم هذه الأحرف المستخدمة في إنشاء القناع يسهل حفظها ونذكرها في الجدول ٨-١ التالي:

جدول ٨-١ أهم الأحرف المستخدمة في إنشاء القناع

العلامة	تستبدل عند الإدخال بـ
#	مسافة خالية أو أي رقم من ٠ إلى ٩ بالإضافة إلى الإشارات + و -
?	حرف أبجدي من A إلى Z أو a إلى z
A	أي حرف أو رقم
&	أي حرف أو خانة خالية

ويوضح الجدول ٨-٢ التالي أمثلة لأقنعة ومعانيها.

جدول ٨-٢ أمثلة على استخدام الأقنعة

القناع (Mask)	معناه	مثال
####	عدد مكون من أربعة أرقام	1234
####-####	رقم تليفون (بالكود الدولي)	(202)123-12345678

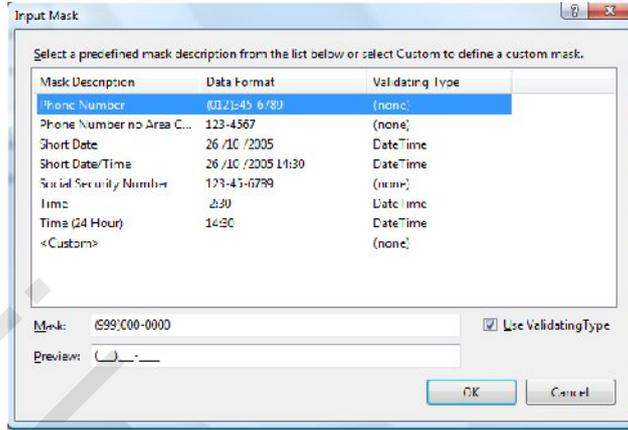
القناع (Mask)	معناه	مثال
#####		
AAA ##,####	تاريخ	Jan 10,2008
## / ##/ ##	تاريخ	10/01/2008
A##:##	توقيت	12:00 P

عند تشغيل البرنامج وقبل إدخال المعلومة إلى مربع الإدخال المقيد يظهر داخل المربع الرمز (_) مكان كل حرف مما يدل على وجود مكان صالح للإدخال. وهذا هو الرمز التلقائي، فإذا أردت أن يظهر بالمربع رمز أو حرف غيره، يمكنك تغييره من خلال الخاصية PromptChar كأن تقوم بجعله # في حالة إدخال أرقام (انظر شكل ٨-٥).



شكل ٨-٥ نموذج لتغيير الحرف الذي يظهر في مربع الإدخال المقيد

- وبدلاً من تعيين قيمة قناع الإدخال بنفسك، يمكنك استخدام أي من الأقنعة المعروفة مسبقاً داخل .Net.. لأداء ذلك، تابع معنا الخطوات الآتية:
١. نشط أداة مربع الإدخال المقيد داخل النموذج.
 ٢. انتقل إلى الخاصية Mask بمربع الخصائص ثم انقر بداخلها.
 ٣. انقر الزر  المجاور للخاصية، يظهر المربع الحوارى Input Mask (انظر شكل ٨-٦).



شكل ٨-٦ تعيين قناع الإدخال من خلال المربع الحوارى Input Mask

٤. اختر القناع المناسب من مربع السرد الموجود بالمربع الحوارى والذي يحتوى بدوره على سبعة أقنعة والتنسيق المصاحب لكل منها، ولاحظ القناع بمربع النص **Mask** بالجزء السفلى من المربع الحوارى وكذلك معاينة القناع كما سيظهر أثناء التشغيل بمربع النص **Preview**.

٥. انقر زر **Ok** لإغلاق المربع الحوارى وإضافة القناع إلى الخاصية **Mask**.

قراءة محتويات مربع الإدخال المقيد **Masked Textbox**

يتم التعامل مع هذا النوع من عناصر التحكم كما يتم التعامل مع مربع النص العادي إلا أنه توجد بعض الاختلافات نوضحها فيما يلي:

عند استخدام الخاصية **Text** فإن القيمة المخزنة هي الأرقام التي أدخلها المستخدم بالإضافة إلى الأقواس والعلامات المخزنة في القناع. فمثلاً عند إدخال رقم التليفون إلى القناع التالي:

#####-#### (####) فإن المستخدم يدخل مثلاً القيمة ١٢٣٤٥٦٧٨٩٠ بينما تخزن الخاصية **Text** القيمة هكذا: ١٢٣٤٥٦-٧٨٩٠ (١٢٣) شاملة الأقواس والشرطة.

وللحصول على المدخلات التي أدخلها المستخدم بدون العلامات الأخرى فإننا نجعل الخاصية **TextMaskFormat** بالقيمة **ExcludePromptAndLiterals**. ويعتبر

استخدام هذه الخاصية مهماً في حالة كون الرقم المدخل سيستخدم في حسابات أخرى وبالتالي لا يمكن إدخال العلامات الفاصلة في الحساب. يوضح شكل ٨-٧ التالي الفرق بين ناتج الخاصيتين لأحد مربعات الإدخال.

شكل ٨-٧ تأثير قيم الخاصية `TextMaskFormat` على الخاصية `Text` المصاحبة لمربع الإدخال المقيد

يمكنك أيضاً استخدام الخاصية `CutCopyMaskFormat` لتحديد النص الناتج عن إجراء عمليتي النسخ والقص داخل مربع الإدخال المقيد بنفس طريقة استخدام الخاصية `TextMaskFormat`.



ويمكنك التأكد من صحة البيانات التي يقوم المستخدم بإدخالها عن طريق تخصيص حالة لهذا النوع من البيانات إلى الخاصية `ValidatingType`، كما يمكنك التعرف على مدى صحة البيانات المدخلة بمجرد انتقال التركيز من مربع الإدخال المقيد إلى أي أداة أخرى من خلال الحدث `TypeValidationCompleted`.

استخدام الخطوط

يتيح نظام `Windows` إمكانيات عديدة لتحسين مظهر النصوص ومن أهمها تغيير الخطوط `Fonts`. والخاصية `Font` توجد في كل العناصر التي تتعامل مع النصوص وهي الخاصية المسؤولة عن الخط المستخدم في عرض النصوص على الشاشة. يوضح شكل ٨-٨ التالي أمثلة مختلفة على استخدام الخطوط.



شكل ٨-٨ أمثلة للخطوط المختلفة التي يمكن استخدامها لتحسين مظهر الإخراج

خصائص الخطوط

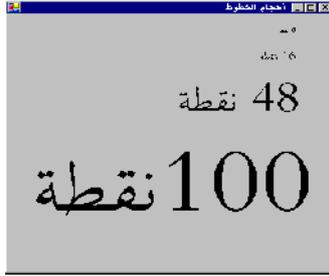
أوضحنا فيما سبق أن الخط رغم أنه خاصية لعناصر مختلفة إلا أنه في حد ذاته كائن مستقل له خصائص تتحكم في كيفية عرض النصوص كما في جدول ٨-٣ التالي.

جدول ٨-٣ خصائص الخطوط

الاستخدام	الخاصية
تحدد اسم الخط المستخدم في عرض النص	Name
تحدد ما إذا كان الخط المراد عرضه ثخين أم عادي	Bold
تحدد هل الأحرف مائلة أم لا	Italic
تحدد هل الأحرف مسطرة بسطر رفيع أسفلها أم لا	Underline
تحدد حجم (بنط) الخط المراد	Size
تحدد عرض الأحرف وعليها خط رفيع يمر بوسطها	StrikeThrough
الوحدة المستخدمة في القياس مثل النقطة Point والبكسل Pixel	Unit
عبارة عن لون الخط	ForeColor

ومن الخواص التي تستخدم غالباً خاصية الحجم الذي يستخدم لزيادة توضيح نص معين في النموذج بتكبيره. وينبغي مراعاة النظام العام للنموذج بحيث لا تنساق وراء الرغبة في الإكثار من الأحجام المختلفة من الخطوط بما يشوه الشكل العام للنموذج وما ينصح به مراعاة حجم الشاشة المستخدمة، فما يستخدم على كمبيوتر محمول يجب أن يكون أكبر في الحجم مما يستخدم على كمبيوتر مكتبي عادي.

الحجم المعتاد للنصوص يتراوح بين ٨ و ٢٠. ولكن يمكنك تحديد الحجم بين ١ و ٢٠٠٠ لو أردت، ولكن لاحظ أن النافذة لن تسع إلا عدد قليل من الأحرف المكتوبة بحجم كبير. يوضح شكل ٨-٩ التالي أمثلة للأحجام المختلفة للخطوط.



شكل ٨-٩ أمثلة لأحجام مختلفة للخطوط

الرقم الذي تدخله في الخاصية **Size** وحداته هي النقطة الطباعية **Point** وهي تساوي $1/72$ من البوصة، أي أنها تساوي ٠.٣٥ مم تقريبا، ويكون الحجم ١٢ موازي لـ ٤.٥ مم تقريبا، و عليه تسع البوصة طويلا حوالي ٦ أسطر من النص المكتوب بهذا الحجم. يمكنك بالطبع تغيير وحدة القياس المستخدمة من الخاصية **Unit** كما ذكرنا.



التحكم في الخطوط في برنامجك

يمكنك التحكم في الخطوط المستخدمة في النموذج كله سواء كان لكل النموذج نفس الخط أو لكل عنصر خط مستقل، ويمكن التحكم في شكل الخطوط في الحالتين، حالة التصميم أو حالة تشغيل البرنامج وقد يسمح البرنامج للمستخدم بإدخال تعديلات على الخطوط المستخدمة.

ضبط الخط الأساسي للنموذج

يستخدم **Visual Basic** خط افتراضي وهو **Tahoma** لأي نموذج تنشئه وإذا لم تقم بتغيير هذا الخط سيستخدمه البرنامج لكل العناصر التي ستضعها على هذا النموذج. ولكي يمكن تنسيق نموذج بحيث يكون كل عناصر هذا النموذج بنفس الخط، فكل ما عليك هو

تغيير الخط المستخدم في النموذج قبل إضافة أي عنصر تحكم إليه مما يؤدي إلى اعتبار هذا الخط افتراضي في كل عنصر يضاف إلى النموذج. لتغيير خط النافذة (أثناء التصميم)، انقر الزر المجاور للخاصية **Font** داخل مربع الخصائص، حيث يظهر المربع الحواري **Font** الذي يمكنك من خلاله تعيين الخط المطلوب ومواصفاته المختلفة (انظر شكل ٨-١٠). يمكنك أيضاً اختيار أحد الخطوط بنقر السهم المجاور للخاصية **Name** داخل المجموعة **Font** بمربع الخصائص ثم اختيار الخط المطلوب من القائمة المنسدلة.



شكل ٨-١٠ اختيار الخط من المربع الحواري **Font**.

تغيير الخط المستخدم في النموذج لا يؤثر في العناصر التي تم وضعها قبل عملية التغيير هذه إذ تظل محتفظة بالخط الخاص بها. أيضاً تغيير الخط المستخدم في النموذج لا يؤثر في الخط المستخدم في شريط العنوان، وإنما يتم تغيير الخط المستخدم في شريط العنوان من لوحة التحكم الخاصة بنظام التشغيل.



ينبغي الحذر من اختيار خطوط قد لا توجد عند المستخدم مما يجعل **Windows** يستبدلها بأقرب الخطوط إليها مما قد يؤدي إلى تشويه الشكل العام لواجهة البرنامج، لذا ننصح دائماً باستخدام خطوط قياسية قدر الإمكان.



بالإضافة إلى تغيير الخط الخاص بالنافذة يمكن تغيير خطوط كل عنصر تحكم على حده سواء أثناء التصميم أو أثناء تشغيل البرنامج. يوضح شكل ٨-١١ استخدام أزرار أوامر لتغيير خصائص الخط في أحد مربعات النصوص.



شكل ٨-١١ تغيير خصائص الخط أثناء تشغيل البرنامج

وفي الإصدارات القديمة من **Visual Basic** والتي تسبق **Visual Basic.Net**، كنا نتعامل مع الخطوط أثناء التشغيل بطريقتين، إما من خلال الكائن **stdFont** أو بتخصيص القيم إلى الخصائص مباشرةً. أما في **Visual Basic 2008** (ومن قبله **Visual Basic.Net** و **Visual Basic 2005**)، فيتم تخصيص القيم إلى خصائص الخط من خلال كائن واحد فقط هو الكائن **System.Drawing.Font** حيث تبقى الخطوط قابلة للقراءة فقط أثناء التشغيل. ففي **Visual Basic 6.0** على سبيل المثال كان من الممكن تخصيص خط وحجم جديدين لمربع النص كما يلي:

```
TextBox1.Font.Name = "Times New Roman"
```

```
TextBox1.Font.Size= 15
```

أما في **Visual Basic 2008** فالأمر يختلف إلى حد ما. حيث يتم تنفيذ الكود السابق كما يلي:

```
TextBox1.Font = New System.Drawing.Font("Arial", 15)
```

كما يمكنك تخصيص المزيد من الخصائص كما يلي:

```
TextBox1.Font = New System.Drawing.Font(TextBox1.Font, FontStyle.Bold)
```

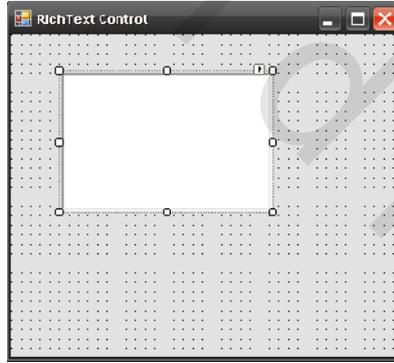

استخدام أداة مربع النص المنوع *RichTextBox Control*

لو وضعت أداة مربع نص منوع (*RichTextBox Control*) بدلاً من مربع نص عادي وسميته بنفس الاسم ثم قمت بتشغيل البرنامج، فلن يحدث أي فرق. فكما قلنا كافة خصائص مربع النص موجودة في الأداة *RichTextBox*. بالإضافة إلى ذلك تدعم هذه الأداة خصائص ووظائف أخرى خاصة بمهمتها.

الطريقة المتبعة في تشكيل النصوص أن يقوم المستخدم أو كود البرنامج بتحديد (تعليم) النص المراد تشكيله، ثم نستخدم الخواص *SelectionFont*, *SelectionColor* وغيرها لتطبيق التنسيق المراد. عند عدم اختيار أي نص فإن الخاصية التي يتم تغييرها سيتم تطبيقها من موضع مؤشر الإدخال تماماً كما يحدث في برامج تنسيق النصوص.

إعداد الأداة *RichTextBox Control*

يمكنك إضافة أداة مربع النص المنوع إلى النموذج بنفس طريقة إضافة الأدوات الأخرى باختيارها من مربع الأدوات ثم رسمها على النموذج (انظر شكل ٨-١٣).



شكل ٨-١٣ مربع النص المنوع عند وضعه على النموذج

تعيين خصائص الخط

تحتوي أداة النص المنوع على عدد من خيارات التنسيق التي تتحكم في ظهور النص داخل الأداة، حيث يمكنك تعيين خصائص الخط الشهيرة مثل **Bold** و **Italic** و **Underline** من خلال الخاصية *SelectionFont* كما يمكنك استخدام نفس الخاصية في تغيير حجم

ومظهر الخط. يمكنك أيضاً تغيير لون الخط المستخدم من خلال الخاصية SelectionColor. وعامةً يمكنك تغيير مظهر النص داخل أداة مربع النص المنوع باتباع إحدى الطرق التالية:

- تعيين الخاصية SelectionFont بالخط المناسب
- تمكين المستخدم من تعيين مواصفات الخط بنفسه من خلال المربع الحوارى Font وذلك باستخدام الأداة FontDialog (راجع الجزء الأول)
- تعيين الخاصية SelectionColor بلون الخط المناسب
- تمكين المستخدم من تعيين لون الخط بنفسه من خلال المربع الحوارى Color وذلك باستخدام الأداة ColorDialog (راجع الجزء الأول)

فإذا أردت على سبيل المثال تغيير خط ولون النص المختار، يمكنك استخدام الكود التالي:

```
RTBox.SelectionFont = New Font("Tahoma", 12, FontStyle.Bold)
RTBox.SelectionColor = Color.Red
```

تنطبق هذه الخصائص فقط على النص المختار داخل مربع النص المنوع. فإذا لم يكن هناك نصاً مختاراً، يتم تطبيق الخصائص على النص الذى يتم كتابته بدايةً من نقطة الإدراج.



تعيين الهوامش والفقرات داخل مربع النص المنوع

يمكنك تعيين هوامش النص داخل مربع النص المنوع كما يمكنك أيضاً إنشاء تعداد نقطى على غرار المستخدم فى برامج معالجة النصوص. لتحويل الفقرات المختارة إلى تعداد نقطى،

قم بتخصيص القيمة True للخاصية SelectionBullet كما يلي:

```
RTBox.SelectionBullet = True
```

يمكنك أيضاً تعيين الهوامش من خلال ثلاث خصائص كما يلي:

- تستخدم الخاصية SelectionIndent لتعيين المسافة بالنقاط بين الحافة اليسرى للأداة والحافة اليسرى للنص.
- تستخدم الخاصية SelectionHangingIndent لتعيين المسافة بالنقاط بين

الحافة اليسرى للخط الأول في الفقرة والحافة اليسرى للخطوط التابعة في نفس الفقرة.

- تستخدم الخاصية **SelectionRightIndent** لتعيين المسافة بالنقاط بين الحافة اليمنى للأداة والحافة اليمنى للنص.

إذا كانت قيمة الخاصية **RightToLeft** الخاصة بالأداة هي **True** فإن الهوامش تكون في الاتجاه المعاكس.



يمكنك تعيين قيم هذه الخصائص أثناء التشغيل كما في الكود التالي:

```
RTBox.SelectionIndent = 8
RTBox.SelectionHangingIndent = 3
RTBox.SelectionRightIndent = 12
```

استخدام أداة النص المنوع في عرض ملفات *RTF*

هناك وظيفتان هامتان لأداة النص المنوع **RichTextBox** وهما **LoadFile()** والتي تقوم بتحميل ملف نصي إلى المربع وعرضه، والوظيفة **SaveFile()** وتستخدم لحفظ محتويات المربع داخل ملف.

صيغة الوظيفة الأولى كالتالي:

```
RTBox.LoadFile ("PathName", filetype)
```

وصيغة الوظيفة الثانية هي:

```
RTBox.SaveFile ("PathName", filetype)
```

حيث:

- **RTBox** هو اسم مربع النص المنوع **RichTextBox**
- **PathName** هو اسم الملف المطلوب والذي يتم تعيينه عملياً من خلال الكائن **OpenFileDialog** (راجع الجزء الأول)
- **FileType** هو رقم يوضح نوعية الملف المراد فتحه من خمس قيم مختلفة هي **RichNoOleObj** و **Rich-Text** و **UnicodePlainText** و **PlainText** و **TextTextObj**.

وذلك كما في الكود التالي:

```
RTBox.LoadFile("c:\myCV.Txt", RichTextBoxStreamType.  
PlainText)  
RTBox.SaveFile("c:\myCV.Txt", RichTextBoxStreamType.  
UnicodePlainText)
```

البحث داخل مربع النص المنوع

رغم إمكانية البحث باستخدام **InStr** داخل الخاصية **Text** لمربع النص المنوع إلا أن هناك وظيفة أكثر كفاءة في البحث وهي الوظيفة **Find** والتي لا تقوم فقط بالبحث، وإنما تقوم أيضاً بتحديد النص في حالة العثور عليه مما يسمح بعد ذلك بتغيير خصائصه بالوظائف المشار إليها سابقاً.

الوظيفة **Find** لها الصيغة العامة الآتية:

```
RTBox.Find( searchstr [, start] [, end] [, options] )
```

في الصيغة السابقة **searchstr** عبارة عن النص المراد البحث عنه. كما ترى بقية المعاملات اختيارية. المعاملان **start** و **end** يحددان مجال البحث وهما ترتيب الحرف الذي سيبدأ والحرف الذي سينتهي عنده البحث على الترتيب. أما المعامل **Options** فيحدد إن كان البحث سيتم مع أخذ حالة الأحرف اللاتينية (كبيرة / صغيرة) في الاعتبار أم لا وخصائص أخرى عند البحث تتضمن ما يلي:

```
MatchCase, NoHighlight, Non, Reverse, WholeWord
```

المثال التالي يبحث عن القيمة الموضوعية في مربع النص **TextBox1** في المربع بأكمله و **MatchCase** تعني أخذ حالة الأحرف في الاعتبار.

```
FoundPos = RTBox.Find("Ali", RichTextBoxFinds.MatchCase)
```

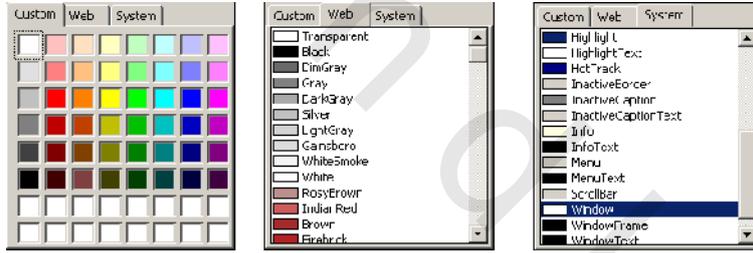
استخدام الألوان

تعتبر الألوان عنصراً هاماً عندما تريد جذب انتباه المستخدم أو تحسين شكل النافذة. وغالباً ما يكون تغيير الألوان خلال طور التصميم أكثر، وتتيح معظم أدوات التحكم تغيير ألوانها من خلال الخاصيتين **ForeColor** للون الواجهة و **BackColor** للون الخلفية.

هناك أسلوبان لتحديد الألوان في البرنامج. الأسلوب الأول أن تحدد الألوان تبعاً لألوان النظام (التي يتم تحديدها من لوحة التحكم في Windows) وبذلك ستتغير الألوان في برنامجك مع تغير ألوان النظام و هذا أفضل عموماً، الأسلوب الثاني أن تحدد الألوان مباشرة بغض النظر عن ألوان النظام، ورغم حريتك هنا إلا أن برنامجك لن يكون متسقاً مع بقية البرامج من حيث الألوان.

استخدام قائمة الألوان في نافذة الخصائص

يمكنك استخدام الخاصيتين **ForeColor** و **BackColor** في نافذة الخصائص لعرض قائمة الألوان التي تحتوي على ثلاث تبويبات (انظر شكل ٨-١٤)، التبويب الأول **Custom** يحتوي على الألوان مباشرة بينما يحتوي التبويب الثاني **Web** على الألوان المستخدمة بكثرة داخل الويب وأخيراً التبويب الثالث **System** الذي يقوم بإظهار الألوان الخاصة بالنظام ويتيح لك تخصيص الألوان تبعاً لألوان النظام.

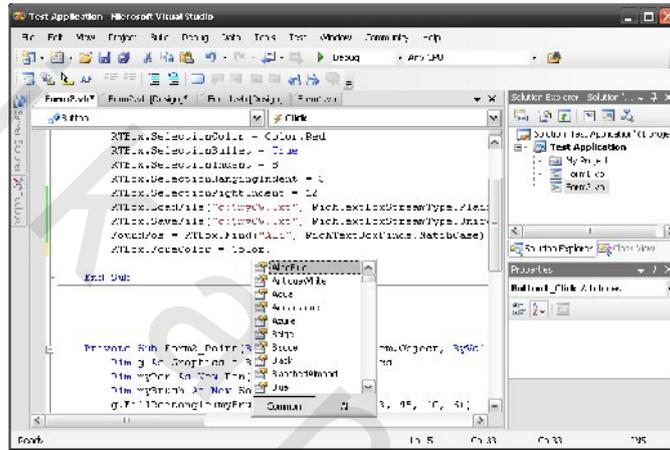


شكل ٨-١٤ تبويبات اختيار الألوان

تغيير الألوان باستخدام الكود

ما سبق كان حديثاً عن تغيير الألوان أثناء التصميم، أما أثناء التشغيل فيجب أن نحدد الألوان باستخدام الخاصية **ForeColor** و **BackColor** ولكن أي القيم سنعطيهما؟ في نظام **Windows** يتم تعريف أي لون من خلال مكوناته من الألوان الأساسية (الأحمر والأخضر والأزرق) وذلك بدمجهم في عدد واحد يعبر عن هذا اللون المختار والذي غالباً ما يكتب بالنظام السداسي عشر، فمثلاً اللون الأزرق يتم التعبير عنه بالرقم **&H00FF0000&**.

ولكن هذا الرقم صعب في التعامل كما ترى، وهناك ثوابت معرفة للألوان الشائعة من الأفضل استخدامها وهي الألوان الموجودة داخل التصنيف **Color**. لذا يمكنك داخل الكود اختيار اللون المطلوب بسهولة تامة. فحينما تكتب **Color** يتم إظهار قائمة بالألوان المتاحة التي تستطيع الاختيار من بينها (انظر شكل ٨-١٥)



شكل ٨-١٥ اختيار اللون المناسب من قائمة الألوان التي تظهر بمجرد كتابة **Color**.

مثال لاستخدام هذه الثوابت:

RTBox.ForeColor = Color.Blue

معنى هذا المثال أن لون أمامية الأداة **RTBox** هو اللون الأزرق.

