بسم الله الرحمن الرحيم

سلسلة دروس
تعلم بالتفصيل خصائص وطرق إستخدام
Dot Net Controls with C#

C# PROGRAMMING

هدية متواضعة لمنتدى الفريق العربي للبرمجة ٢٠٠٠

www.arabteam2000.com

Senior Developer : Ahmed H. Alawady
Mobil : +2-012-3456808 / Tel : +2-055-3661835 Email : Alawady_ahmed@hotmail.com

السلام عليكم ورحمة الله وبركاته

هنا سوف يتم إن شاء الله شرح كل ما يتعلق بالأدوات الخاصة بـالدوت نت مع شرحها خاصية خاصية Propery وبطرق إستخدام الأحداث Events وفائدة كل أداه من الأدوات وسوف نمر بالأدوات كما هي بالترتيب الأبجدي للأدوات وهي كالآتي:

1		Button	2		CheckBox
3		CheckedListBox	4		ColorDialog
5		ComboBox	6		ContextMenu
7		DataGrid	8		DateTimePicker
9		DomainUpDown	10		ErrorProvider
11		FontDialog	12		GroupBox
13		HScrollBar	14		ImageList
15		Label	16		LinkLabel
17		ListBox	18		ListView
19		MainMenu	20		NumericUpDown
21		OpenFileDialog	22		PageSetupDialog
23		Panel	24		PictureBox
25		PrintDialog	26		PrintDocument
27		PrintPreviewControl	28		ProgressBar
29		PrintPreviewDialog	30		CrystalReportViewer
31		RadioButton	32		RichTextBox
33		SaveFileDialog	34		StatusBar
35		TabControl	36		TextBox
37		Timer	38		ToolBar
39		ToolTip	40		TrackBar
41		TreeView	42		VScrollBar
43		MonthCalender	44		Spliiter
45		HelpProvider	46		NotifyIcon

وبالطبع قبل ذلك سوف أقوم بشرح إن شـاء الله الخـصائص المـشتركة لـبعض الأدوات لكـي لا نقـوم بـالتكرار بعـد ذلـك . نـسألكم الـدعاء بـالتوفيق ونـسأل الله العـون فـي هـذا لمـا فيـه خيـر الإسـلام والمـسلمون.

Senior Developer : Ahmed H. Alawady
Mobil : +2-012-3456808 / Tel : +2-055-3661835 Email : Alawady_ahmed@hotmail.com

1

AccessibleDescription Property

وهي بالصيغة التالية

```
public string AccessibleDescription {get; set;}
```

الفائدة : يقوم بإرسال وإستقبال وصف للأداه المستخدمة ، وذلك للمستخدم ذوي الإحتياجات الخاصة .

أمثلة للإستخدام : عند وجود مثلا StatusBar Control وتريد عند الوقوف على أي كنترول بالشاشة يتم شرح فائدة الكنترول الحالية في StatusBar فبالطبع نقوم بإرسال الوصف المخزن في الخاصية AccessibleDescription

وبالمثل يمكن التفكير في كيفية إستخدام توصيف الكنترول المستخدم الذي يقف عليه المستخدم أيضا من الممكن عمل زر بصورة علامة إستفهام بحيث عند الضغط عليه ثم الضغط على الكنترول المراد الإستفسار منه يقوم بعرض الوصف الذي قمنا بتخزينة سابقا في الخاصية AccessibleDescription .... ألخ

طريقة الإستخدام

```
System.Windows.Forms.TextBox txt = new System.Windows.Forms.TextBox();
// هذه هي فائدة إستخدام Set
txt.AccessibleDescription = "العربية هذه الخانة يتم كتابة إسم الشركة باللغة";
وبعد ذلك في وسط البرنامج
// هذه هي فائدة إستخدام Get
string textBox_Desc = txt.AccessibleDescription;
```

2

Property : AccessibleName

وهي بالصيغة التالية

```
public string AccessibleName {get; set;}
```

شرح الأخ الفاضل معتز شمس : فان الـ AccessibleName و الـ AccessibleDescription وبعض الخواص الأخرى المشابهه ... لهم أهميه كبيره لأنه يتم استخدامهم من قبل البرامج والأدوات الخاصه بالـ Accessibility مثل برنامج Microsoft Narrator.

مثلا برنامج الـ Narrator مفيد للمستخدمين ضعفاء النظر جدا أو العمى والذين يودون استخدام الكمبيوتر والتعامل معه.

فبرنامج الـ Narrator الذي ينزل مع الـ Windows يقوم بنطق مايحدث على الشاشه وبالأخص يقوم بنطق تلك الكلمات الموجوده في خاصية AccessibleName وبالاضافه الى غيرها ربما. (النسخه الموجوده لدى مع Windows Server 2003 لا تنطق الا الموجود في AccessibleName فقط).

مثلا اذا كان لدينا هنا Form لعمل Login تتكون من ٢ TextBoxes لادخال اسم المستخدم وكلمة المرور ويوجد لدينا Button اسمه مثلا OK ... فاذا كان مستخدم البرنامج يعاني من مشاكل في النظر أو أعمى تماما فينبغي عليك اضافة بعض البيانات لتلك الـ Controls حتى يستطيع برنامج الـ Narrator أن ينطقها للمستخدم.

مثلا في الـ Login Form نجعل الـ AccessibleName يحتوي على ::

```
ourLoginForm.AccessibleName = "Login To ArabTeam Form";
```

وأيضا الـ TextBoxes ::

```
txtBoxUserName.AccessibleName = "Your User Name Please";
txtBoxPassword.AccessibleName = "Your Password Please";
```

وفي الـ Button ::

```
okButton.AccessibleName="Press this button to login";
```

فعندما يفتح المستخدم برنامجنا هذا وهو بالفعل مشغل برنامج الـ Narrator ويغير الـ Focus من خلال زرار Tab أو الأسهم ... تجد الـ Narrator ينطق له كل ما كتبناه داخل الـ AccessibleName الخاصه بالـ Control

المنشطه (Got Focus).

بالنسبه أيضا لـ AccessibleDescription وماشابهها فيتم استخدامها من برامج أخرى من أمثال الـ Narrator.

الفائدة : يقوم بإرسال وإستقبال وصف للأداه المستخدمة

وهي خاصية مساعدة مثل الخاصية السابقة

ولكن فائدتها هي إسم وصف للأداه وليس وصف لها (ملخص لإسم الكنترول)

مثال توضيحي : عندما يكون لديك كنترول يعرض صورة فيمكنك كتابة إسم الخاصية AccessibleName مثلا بكلمة Preview

طريقة الإستخدام :

```
System.Windows.Forms.TextBox txt = new System.Windows.Forms.TextBox();
// هذه هي فائدة إستخدام Set
txt.AccessibleName = "CompanyName";
txt.AccessibleDescription = "الخانة يتم كتابة إسم الشركة باللغة العربية هذه";
وبعد ذلك في وسط البرنامج
// هذه هي فائدة إستخدام Get
string textBox_Name = txt.AccessibleName;
string textBox_Desc = txt.AccessibleDescription;
```

3

Property : AccessibleRole

وهي بالصيغة التالية

```
public AccessibleRole AccessibleRole {get; set;}
```

الفائدة : يقوم بإرسال وإستقبال دور والمرونة التي يؤديها الكنترول للمستخدم ذوي الإحتياج الخاص بالطبع

Senior Developer : Ahmed H. Alawady
Mobil : +2-012-3456808 / Tel : +2-055-3661835 Email : Alawady_ahmed@hotmail.com

وهي خاصية مرتبطة بالخاصيتين السابقتين في المهمة لتسهيل البرنامج لذوي الإحتياج الخاص

مثال توضيحي : وهذه الخاصية AccessibleRole تقوم بوصف نوع مستخدم الكنترول
وهي من نوع Enumeration ومن ضمن المجموعة التي تتضمنها (Alert) وهي ميزة فائدتها أنها تقوم بعمل صوت لتنبية المستخدم عن هذا الكنترول
شرح الأخ الفاضل معتز شمس : فهذه الخاصيه يمكنك اسناد اليها احدى القيم الموجوده في AccessibleRole enum .... وهذه الخاصيه توصف للمستخدم ذو الاحتياجات الخاصه كيفية التعامل مع الـ Control وكيف يستخدمها ... أى بمعنى انه ان كان هذه الـ Contorl النشطه من نوع Button فان الـ Narrator مثلا يقرأ له الـ Text الموجود في الخاصيه AccessibleName كما ذكرت في السابق بجانب أيضا أن يخبره عن كيفية استعمال الـ Button .. بأن يخبره مثلا أن يضغط على Space.

الخاصيه AccessibleRole دائما تكون = AccessibleRole.Default ولا ينصح تغييرها ، فاتركها دائما كما هى Default حتى ينطق الـ Narrator الطريقه الصحيحه عن كيفية الضغط والتعامل مع الـ Controls.

غيرها فقط في بعض الحالات .. منها اذا كنت تريد تشتيت المستخدم (حرام عليك يا مفتري) وتريد مثلا أن تجعل الـ Narrator ينطق للمستخدم أن الـ Button يتم الضغط عليه باستخدام الأسهم !!! أو تغيرها اذا كنت من هواة كتابة الكود دائما بطريقه صريحه .. مثلا في الـ Button تكتب ::

```
okButton.AccessibleRole = AccessibleRole.PushButton;
```

وفي الـ Form تكتب ::

```
ourLoginForm.AccessibleRole = AccessibleRole.Window;
```

أيضا من الحالات التي ينبغي عليك أن تحددها بطريقه صريحه ولا تجعلها Default اذا كنت تقوم بعمل Control خاصه بك ... فمثلا اذا كنت صممت أنت Control جديده مشابهه للـ ComboBox فكيف للدوت نت أن تعلم كيفية استخدام أداتك اذا قمت بعمل AccessibleRole.Default لها ؟

ولذلك يجب أن تحدد كيفية استخدام أداتك بطريقه صريحه حتى تجعل الـ Narrator يخبر المستخدم عن كيفية استخدامه للأداتك حين تكون نشطه .. فاذا كانت أداتك مثلا تشبه الـ ComboBox من ناحية التعامل معه فقم بكتابة مثلا ::

```
yourUserControl.AccessibleRole = AccessibleRole.ComboBox;
```

أو على حسب كيفية استخدام أداتك .. وتصفح AccessibleRole enum جيدا بعد عمل الـ Control الخاصه بك أو عند استخدامك لأداه خارجيه غير موجوده في الدوت نت حتى تعطي AccessibleRole مناسبه.

4

Property : AllowDrop

وهي بالصيغة التالية

```
public virtual bool AllowDrop {get; set;}
```

AccessibleRole	Default
AllowDrop	True
AutoScale	True

الفائدة : تقوم بإرسال وإستقبال قيمة لتدل على أن الكنترول يسمح أو لا يسمح بعمل بيانات يقوم المستخدم بسحبها ووضعها في الكنترول

وبالطبع القيم التي تأخذها هذه الخاصية هي True أو False

مثال توضيحي : فرضا لديك كنترول يقوم بإستقبال وعرض صورة وتريد المستخدم أن يقوم بسحب الصورة من ملف من على جهاز الكمبيوتر ووضعة داخل الكنترول المخصص بالصورة ، ثم يقوم الكنترول بإستقبال هذه الصورة المسحوبة ثم عرضها ويمكننا بالطبع إستخدامها بعد ذلك في حفظها مثلا .

طريقة الإستخدام :

```
PictureBox pic = new PictureBox();
pic.AllowDrop = true;
or
bool HasDragDrop = pic.AllowDrop;
```

يوجد مثال عملي ممتاز لسحب صورة ووضعها داخل كنترول PictureBox على هذا الرابط

A Simple Drag And Drop How To Example

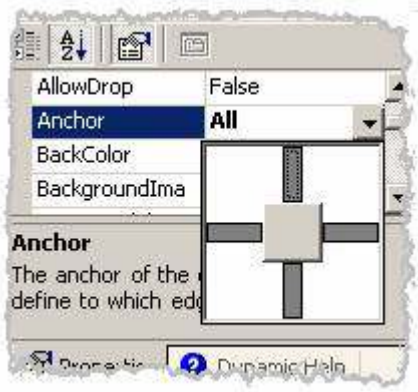
أو مباشرة من على الرابط التالي : http://www.codeproject.com/csharp/dandtutorial.asp

5

Property : Anchor

وهي بالصيغة التالية

```
public virtual AnchorStyles Anchor {get; set;}
```



الفائدة : يقوم بإرسال وإستقبال حدود الكنترول المثبتة لحدود (الحواف) للكنترول الذي يحتوية ، لتكبير وتصغير حجم الكنترول على أساس تكبير وتصغير الكنترول الأب الذي يحتوية بحيث يعتمد على نوع الحواف Edgesالمرتبطة مع بعضها البعض .

مثال توضيحي :

عند تثبيت المسافة بين الحافة للكنترول مع الحافة للأب من الجهة المحددة ، أي إذا كان كان التحديد من جهة اليمين وكانت المسافة بينه وبين الأب من جهة اليمين X فعند تكبير الكنترول الأب يقوم بتكبير الكنترول المستخدم ليبقى على نفس المسافة بينه وبين الأب من الحافة اليمنى بمقدار X

طريقة الإستخدام :

```
TextBox txt = new TextBox();
txt.Anchor = System.Windows.Forms.AnchorStyles.Right;
```

حافة وإن كان الكنترول مثبت في أكثر من

Senior Developer : Ahmed H. Alawady
Mobil : +2-012-3456808 / Tel : +2-055-3661835 Email : Alawady_ahmed@hotmail.com

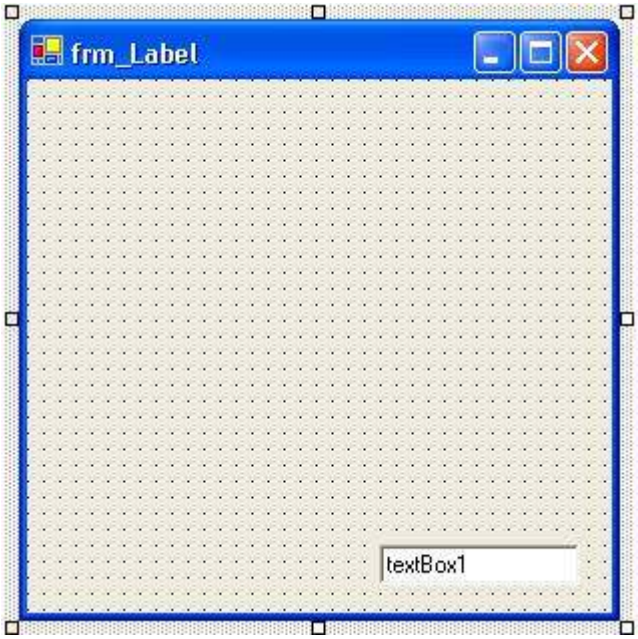
```
txt.Anchor =
((System.Windows.Forms.AnchorStyles)((System.Windows.Forms.AnchorStyles.Bo
ttom | System.Windows.Forms.AnchorStyles.Right)));
```

وذلك على أساس أن قيم الخاصية Anchor موجودة في Enum الخاص

System.Windows.Forms.AnchorStyles

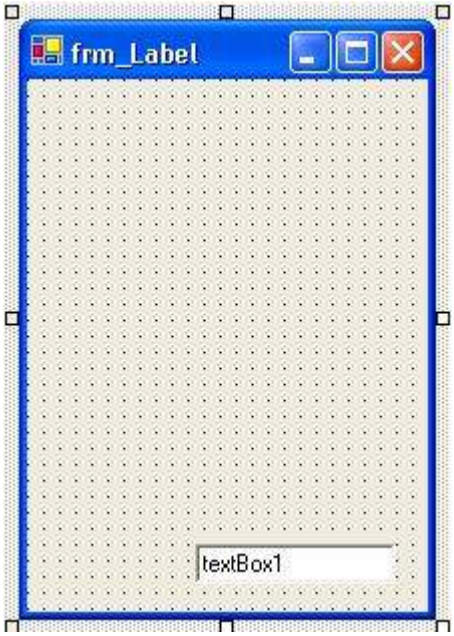
وهي كالتالي :

```
System.Windows.Forms.AnchorStyles {Top, Bottom, Left, Right}
```



ثم عند التصغير من جهة اليمين يحدث التالي

Senior Developer : Ahmed H. Alawady
Mobil : +2-012-3456808 / Tel : +2-055-3661835 Email : Alawady_ahmed@hotmail.com



الفائدة : يقوم بإرسال وإستقبال حدود الكنترول المثبتة لحدود (الحواف) للكنترول الذي يحتوية ، لتكبير وتصغير حجم الكنترول على أساس تكبير وتصغير الكنترول الأب الذي يحتوية بحيث يعتمد على نوع الحواف Edges المرتبطة مع بعضها البعض .

وهذه الحالة عندما يتم عمل تثبيت من جهتين متضادتين أي مثلا من اليمين واليسار ، فعند التصغير من جهة اليمين فإن المسافة من اليمين ثابتة وأيضا اليسار فلذلك يقوم بتصغير حجم الكنترول لكي يترك المسافات ثابتة من الجهتين

فإن لم يتم تثبيتها من الجهتين يتم تحريك الكنترول ولا يتم تصغيرة أو تكبيرة بحيث أنه مثبت من جهة واحدة (كما هو موضح في الصورة السابقة)

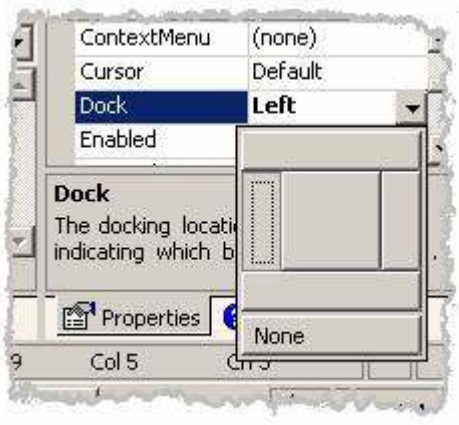
6

Property : Dock

وهي بالصيغة التالية

```
public virtual DockStyle Dock {get; set;}
```


Senior Developer : Ahmed H. Alawady
Mobil : +2-012-3456808 / Tel : +2-055-3661835 Email : Alawady_ahmed@hotmail.com



الفائدة :

تقوم الخاصية بإرسال وإستقبال أي الحدود الخاصة بالكنترول الأب (مثال : إذا كان لدينا TextBox بداخل Form أو بداخل GroupBox فكلا من Form أو GroupBox يقال له كنترول الأب) ليقوم بتثبيتها على حدود الكنترول .

مثال توضيحي :

إذا كان لدينا TextBox موجود بداخل Form لنقوم بإنشاء مثلا برنامج Notbad نلاحظ أن TextBox مثبتة أركانة مع أركان Form السفلى من جهة اليمين واليسار أيضا وبالمثل من أعلى اليمين واليسار ، وإذا قمنا بتمديد (تكبير) Form يقوم TextBox بالتمديد معة وبالعكس في حالة التقلص للـ Form .

وهو بخلاف الخاصية Anchor بحيث أن الخاصية Dock يأخذ بعد X والبعد Y للفورم ثم ينسخها للكنترول TextBox أي أنهما متطابقين في البعد ، أما بالنسبة للخاصية Anchor فإنها تترك المبرمج يترك مسافة ثابتة بين الحافتين TextBox و Form .

يوجد في الخاصية Dock خمسة إختيارات للتثبيت :

١- تثبيت من جهة اليمين

٢- تثبيت من جهة اليسار

٣- تثبيت من جهة الأعلى

٤- تثبيت من جهة الأسفل

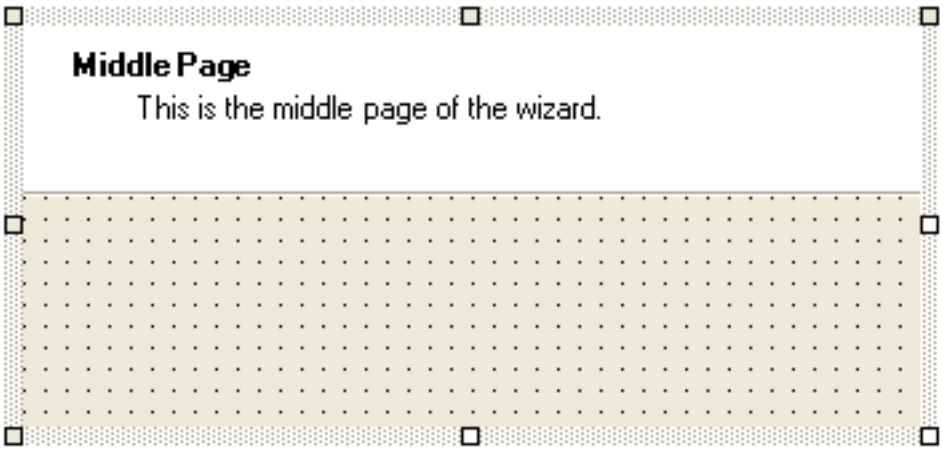
٥- تثبيت عن طريق الملئ

بالنسبة لجهة اليمين : فهو يقوم بتثبيت الكنترول في زاويتي الكنترول الأب من (الركن أعلى اليمين ،

الركن أسفل اليمين) →

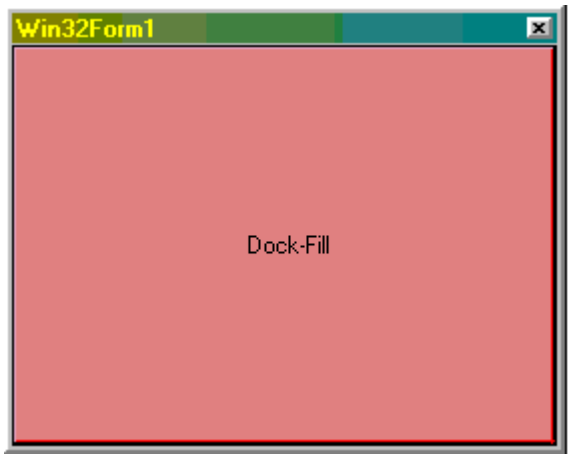
بالنسبة لجهة اليسار : فهو يقوم بتثبيت الكنترول في زاويتي الكنترول الأب من (الركن أعلى اليسار ، الركن أسفل اليسار) ←

بالنسبة لجهة الاعلى : فهو يقوم بتثبيت الكنترول في زاويتي الكنترول الأب من (الركن أعلى اليمين ، الركن أعلى اليسار) ↑



بالنسبة لجهة الأسفل : فهو يقوم بتثبيت الكنترول في زاويتي الكنترول الأب من (الركن أسفل اليمين ، الركن أسفل اليسار) ↓

بالنسبة للملئ : فهو يقوم بتثبيت الكنترول في زاويتي الكنترول الأب من جميع الأركان



مثال لجهة الأسفل (StatusBar) ، وللأعلى (ToolBar أو Taskbar الخاص بالويندوز)

طريقة الإستخدام :

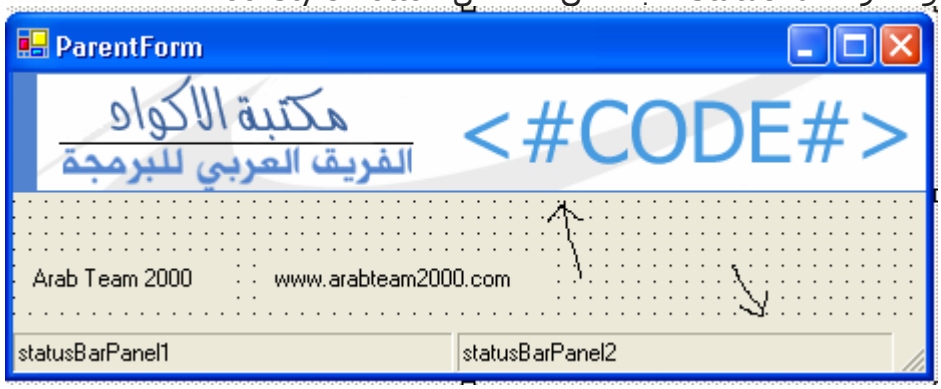
الخاصية تأخذ قيمها من Enum الخاصة بـ DockStyle

```
DockStyle { None, Left, Right, Top, Buttom, Fill }
```

أما بالنسبة لطريقة إستقبال القيمة

```
my_TextBox.Dock = DockStyle.Left;
```

الصورة التالية بها شكل فورم وبها صورة مثبتة من الأعلى DockStyle.Top
والأخرى StatusBar مثبتة من الأسفل DockStyle.Botton

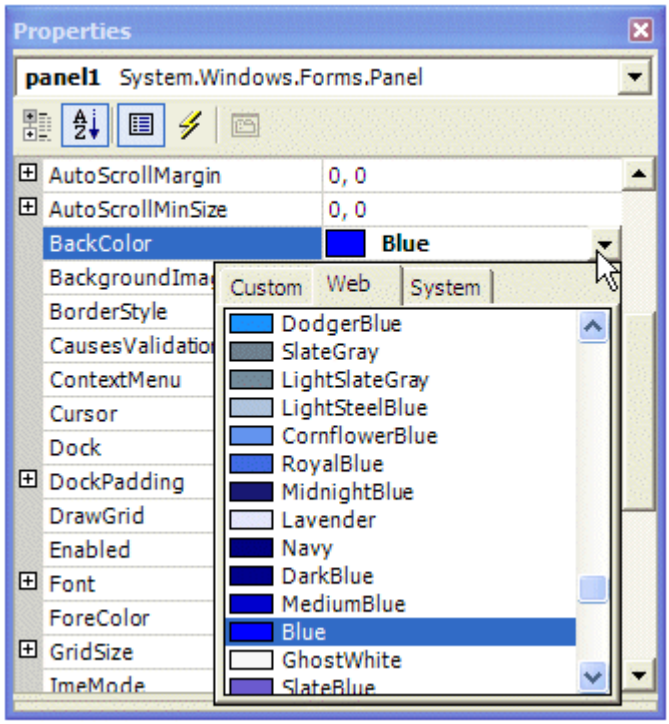


7

Property : BackColor

وهي بالصيغة التالية

```
public virtual Color BackColor {get; set;}
```



الفائدة :

يقوم بإرسال وإستقبال لون الأرضية للكنترول

مثال توضيحي :

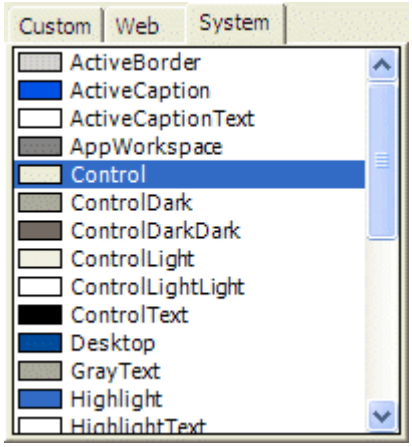
عند إستخدام أداه مثل GroupBox ونريد أن نجعل له أرضية بلون أزرق مثلا ، وبالطبع عندما نريد نفس الفكرة لأرضية الفورم Form

طريقة الإستخدام :

عند إرسال قيمة اللون

```
myControlName.BackColor = System.Drawing.SystemColors.ControlLightLight;
```


Senior Developer : Ahmed H. Alawady
Mobil : +2-012-3456808 / Tel : +2-055-3661835 Email : Alawady_ahmed@hotmail.com



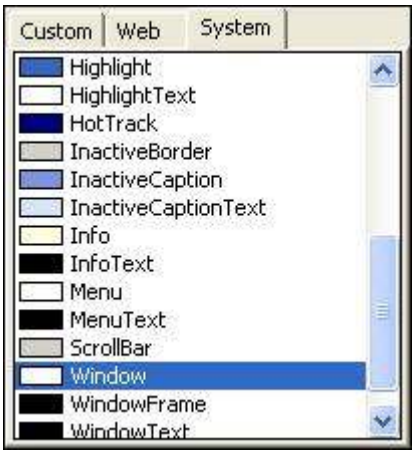
وعند إستقبال اللون من الكنترول

```
System.Drawing.Color myColor = myControlName.BackColor;
```

ويمكن إرسال اللون بأكثر من طريقة :

١- System Colors ويمكن إستخدام الألوان الخاصة بها في Enum الخاص بـ System.Drawing.SystemColors

```
= System.Drawing.SystemColors.ControlLightLight;
```



٢- يمكن أعطاءة رقم اللون المكون من ثلاثة ألوان رئيسية (أحمر ، أخضر ، أزرق)

```
= System.Drawing.Color.FromArgb(((System.Byte)(241)), ((System.Byte)(246)),
((System.Byte)(255)))
```


Senior Developer : Ahmed H. Alawady
Mobil : +2-012-3456808 / Tel : +2-055-3661835 Email : Alawady_ahmed@hotmail.com

٣- Colors ويمكن إستخدام الألوان الخاصة بها في Enum الخاص بـ Colors

```
= System.Drawing.Color.Black;
```

ملحوظة مهمة : أن BackColor لا يستقبل اللون الشفاف Transparent إذا كان الخاصية ControlStyles.SupportsTransparentBackColor للكنترول قيمتها False

والكنترول في الأغلب يأخذ لون الأرضية الخاصة بالكنترول الأب ، فمثلا إن كان أرضية الفورم لونها أحمر فإن الكنترول يأخذ نفس الأرضية

8

Property : ForeColor

وهي بالصيغة التالية

```
public virtual Color ForeColor {get; set;}
```

الفائدة :

يقوم بإرسال وإستقبال لون خط الكتابة الموجود بداخل الكنترول

مثال توضيحي :

عند إستخدام أداه مثل TextBox ونريد أن نجعل لون الكتابة بداخلة حمراء مثلا أو بأي لون آخر

Senior Developer : Ahmed H. Alawady
Mobil : +2-012-3456808 / Tel : +2-055-3661835 Email : Alawady_ahmed@hotmail.com

طريقة الإستخدام :

عند إرسال قيمة اللون

```
myControlName.ForeColor = System.Drawing.SystemColors.ControlLightLight;
```

وعند إستقبال اللون من الكنترول

```
System.Drawing.Color myColor = myControlName.ForeColor;
```

ويمكن إرسال اللون بأكثر من طريقة :

١- System Colors ويمكن إستخدام الألوان الخاصة بها في Enum الخاص بـ System.Drawing.SystemColors

```
= System.Drawing.SystemColors.ControlLightLight;
```

٢- يمكن أعطاءة رقم اللون المكون من ثلاثة ألوان رئيسية (أحمر ، أخضر ، أزرق)

```
= System.Drawing.Color.FromArgb(((System.Byte)(241)), ((System.Byte)(246)),
((System.Byte)(255)))
```

٣- Colors ويمكن إستخدام الألوان الخاصة بها في Enum الخاص بـ Colors

```
= System.Drawing.Color.Black;
```

```
Property : Enabled
```

وهي بالصيغة التالية

```
public bool Enabled {get; set;}
```

الفائدة :

يقوم بإستقبال وإرسال قيمة فعالية الكنترول إما True أو False ، وهو مدى إستجابة الكنترول للمستخدم

مثال توضيحي :

إذا كانت القيمة الخاصة بها True وهي القيمة الإفتراضية يعني أن الكنترول مسموح الإستخدام من قبل المستخدم

وإن كانت القيمة الخاصة بها False يعني أن الكنترول ليس مسموح للمستخدم إستخدامة

فإذا كان لدينا TextBox ولا نريد المستخدم ان يكتب به أي شئ نجعل القيمة الخاصة بالكنترول False وبالطبع العكس صحيح

طريقة الإستخدام :

```
myControl.Enabled = false;
```

وأيضا لمعرفة إن كان فعال أم لا

```
if( myControl.Enabled == false )
```

أو بالمثل للمثال السابق

```
if( !myControl.Enabled )
```

ملحوظة هامة : إذا كانت فعالية الكنترول الأب مثل GroupBox مقفولة (false) يتم قفل وعدم فعالية جميع الكنترولس الموجودة بداخلة

عدم فعالية الكنترول (false) يقوم بعمل إغلاق (عدم تفعيل) لجميع الأحداث الخاصة بالكنترول ، أي إذا كان الكنترول Button وأوقفنا فعاليتة لا يقوم بتفعيل الحدث الخاصة بالضغط عليه OnClick

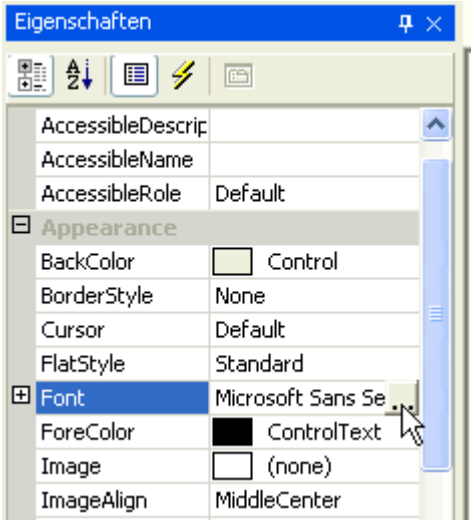
10

Property : Font

وهي بالصيغة التالية

```
public virtual Font Font {get; set;}
```


Senior Developer : Ahmed H. Alawady
Mobil : +2-012-3456808 / Tel : +2-055-3661835 Email : Alawady_ahmed@hotmail.com



الفائدة :

تقوم بإرسال (Get) وإستقبال(Set) كلاس Class من نوع Font ، وذلك لعرض نوع الخط للكلمة المعروضة

مثال توضيحي :

إذا كان لدينا فرضا TextBox وبه كلمة نريد أن نجعل لها حجم خط معين ونوع خط معين ، ومن الممكن أن نجعلة عريض أو مائل ... الخ

وكذلك بقية Controls التي تحتوي على الخاصية نفسها Font

أما بالنسبة للون الخط فهو في الخاصة التالية المسمى ForeColor وسوف نقوم بشرحها إن شاء الله

والقيمة الأفتراضية لهذه القيمة للخاصية هي DefaultFont هو نوع خط إفتراضي في حالة عدم تغيير نوع الخط

وهذه الخاصية أيضا تقوم بأخذ نفس قيمة الخاصية Font للكنترول الأب (Control Parent)

وبالطبع أفضل طريقة ليتمكن المستخدم من تغيير خصائص الخط للكنترول هي أن نقوم بعرض له شاشة تغيير الخطوط FontDialog

ومن ثم إستقبال القيمة المرتجعة ثم إرسالها للكنترول المراد تغيير الخط الخاص به . وسوف نقوم إن شاء الله بشرح الكنترول FontDialog من ضمن القائمة الموجودة في أول الموضوع .

طريقة الإستخدام :

ونقوم بإنشاء متغير حرفي string لتعريف نوع الخط

```
string _FontName = "Kartika";
```

ثم نقوم بإنشاء متغير كسري Float لتغير حجم الخط

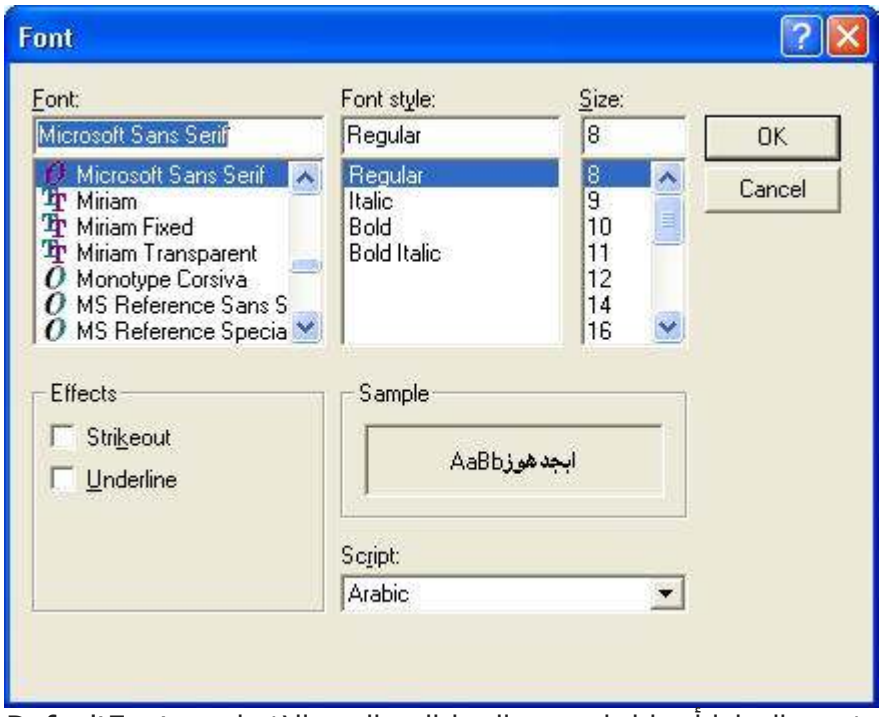
```
float _FontSize = 15F;
```

ثم نقوم بإنشاء متغير Style لتحديد شكل الخط (عريض - مائل - تحته خط)

وفي الحالية التالية نقوم بإختيار النوع عريض ومائل في نفس الوقت

```
System.Drawing.FontStyle _FontStyle =
((System.Drawing.FontStyle)((System.Drawing.FontStyle.Bold |
System.Drawing.FontStyle.Italic)));
```

```
myTextBox.Font = new System.Drawing.Font(_FontName, _FontSize,
_FontStyle);
```

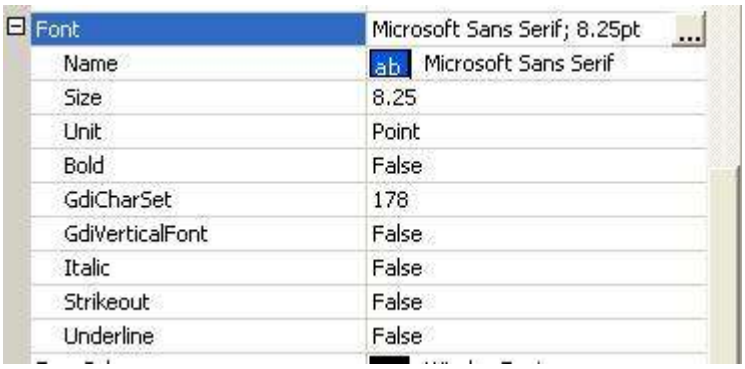


وفي حالة إذا أردنا إعادة نوع الخط إلى النوع الإفتراضي DefaultFont

نقوم بذلك

```
myTextBox.Font = TextBox.DefaultFont;
```

حيث أن TextBox هو Class ، أي أن إذا كان لدينا الكنترول MyButton يكون الكلاس بالطبع Button

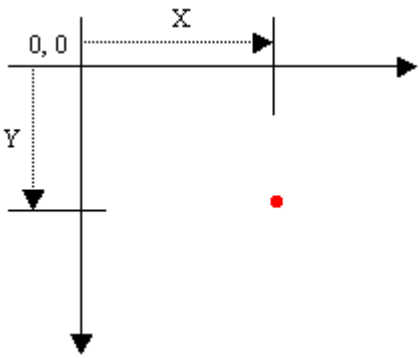


11

Property : Location

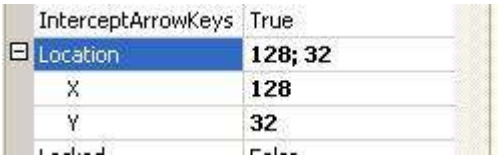
وهي بالصيغة التالية

```
public Point Location {get; set;}
```



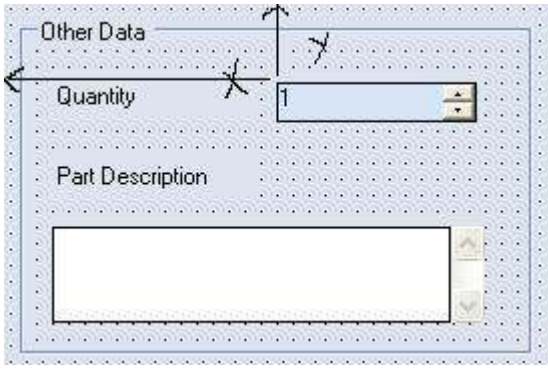
الفائدة :
فائدتها إرسال وإستقبال موقع الكنترول (الحد أعلى اليسار Upper Left) بالنسبة للحد الأب للكنترول (أيضا الحد أعلى اليسار)

مثال توضيحي :
عندما يكون لدينا كنترول بداخل Form ونريد أن نضعه في مكان يبعد عن الحد الأعلى للفورم بمقدار ١٠ وعن الحد الأيسر بمقدار ٢٠
وذلك يفيد في ترتيب وضع الكنترول في الفورم



وعندإستخدام Design Mode لا نهتم بهذه الخاصية لأننا نستخدم الماوس في تحريك الكنترول ، ولكن تتضح أهميتها بالضبط عندما لا يقوم الماوس بوضعها في البعاد الصحيحة التي نريدها وأيضا تتضح أهميتها عند إنشاء كنترول بالكود ونريد تحديد مكانة عند عرضة
وأيضا تتضح أهميتها عندما نريد تحريك الكنترول من مكان لآخر فيمكننا زيادة أو نقصان بعد الكنترول

ويحتوي بعد الكنترول على بعدين فقط X، Y بحيث أن البعد X هو بعد الكنترول من جهة اليسار ، والبعد Y هو بعد الكنترول عن جهة الأعلى.



حيث أن الصورة هي عبارة عن بعد حافة الكنترول عن الحافة للكنترول الأب (نلاحظ أن الكنترول الأب هو GroupBox وليس Formولكن الكنترول الأب للـ GroupBox هو Form هذا في حالة لو كان الكنترول GroupBox لا يحتوية كنترول آخر).

طريقة الإستخدام :

هذا لإعطائها موضع الكنترول

```
myControl.Location = new System.Drawing.Point( 10, 30 );
```

وهذا لمعرفة موضع الكنترول

```
int x = myControl.Location.X;
int y = myControl.Location.Y
```

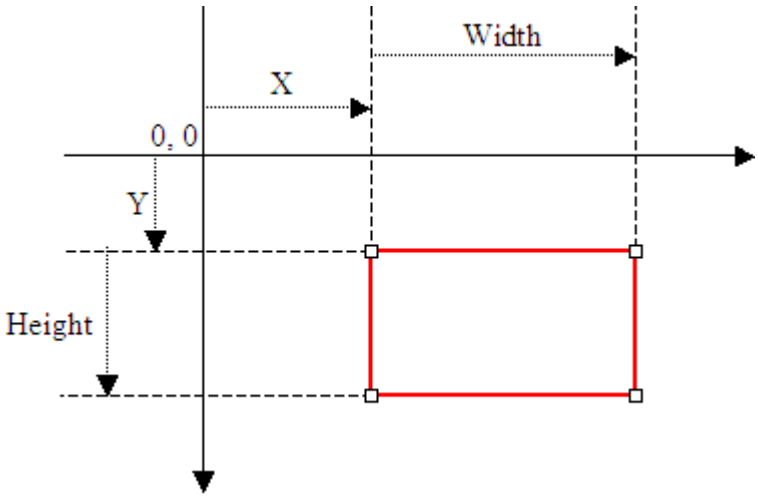

Senior Developer : Ahmed H. Alawady
Mobil : +2-012-3456808 / Tel : +2-055-3661835 Email : Alawady_ahmed@hotmail.com

12

Property : Size

وهي بالصيغة التالية

```
public Size Size {get; set;}
```



الفائدة :

فائدتها إرسال وإستقبال حجم (طول وعرض) الكنترول ، وهي من النوع System.Drawing.Size وهي تحتوي على Width و Height وهما من النوع int

مثال توضيحي :

عندما نضع أي كنترول داخل فورم نقوم بتحديد حجم الكنترول لكي يتنظم شكل الفورم ، ويوجد بالطبع قيمة إفتراضية لحجم الكنترول فعند وضع الكنترول يقوم بوضع القيمة الإفتراضية تلقائيا حتى يقوم المبرمج بتغيير حجمها .

وتوجد أحيانا أوقات نقوم بتكبير وتصغير حجم الفورم Form أثناء عمل البرنامج فنقوم بإستخدام الخاصية هذه .

فعندما يوجد لديك فورم وبه بيانات أساسية وبيانات فرعية بحيث عندما يقوم المستخدم بكتابة البيانات الأساسية يتم توسيع حجم الفورم لإظهار البيانات الفرعية ، وتظهر هذه الخاصية في الشاشة الخطأ للدوت نت عندما ينبهك بوجود خطأ وعندما تريد معرفة المزيد من المعلومات يتم الضغط على الزر Details .

طريقة الإستخدام :

هذا لإعطائها موضع الكنترول

```
myControl.Size = new Size(75,25);
```

وهذا لمعرفة موضع الكنترول

```
int width = myControl.Size.Width;
```

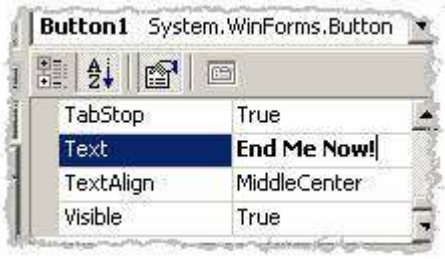
```
int height = myControl.Size.Height;
```

13

Property : Text

وهي بالصيغة التالية :

```
public virtual string Text {get; set;}
```



الفائدة :

هو الإسم أو الجملة المرافق للكنترول وهو بخلاف الخاصية Name ، ويستفاد منها في الكنترول التي تظهر الإسم مثل TextBox، Button وغيرها ولكن لم أجد لها أي فائدة في الكنترول الأتية TreeView، GroupBox لأنهم لا يظهرون الإسم .

مثال توضيحي :

أبسط مثال هو عندما تريد وضع زرين من الكنترول Button ، أحدهما للسؤال عن تنفيذ عملية والآخر لإلغاء التنفيذ فنكتب في الأولى (موافق) والآخر (غير موافق) .

إذا نستنتج من هذه الخاصية أنها هي الإسم أو النص الذي يظهر للمستخدم لهذه الشاشة .

Senior Developer : Ahmed H. Alawady
Mobil : +2-012-3456808 / Tel : +2-055-3661835 Email : Alawady_ahmed@hotmail.com

ففي هذه الصورة يتم توضيح المعنى السابق للخاصية لكل من :

١- الفورم نفسها فقيمة الخاصية Text هي "Ask.."

٢- الكنترول Label الموجود في وسط الشاشة فقيمة الخاصية Text هي "Are you sure ؟"

٣- الزر Button الموجود بجهة اليسار قيمتها "Cancel"

٤- الزر Button الموجود بجهة اليمين قيمتها "Ok"

طريقة الإستخدام :

بما أنها تقوم بإرسال وإستقبال قيمة (Set & Get)

```
myControl.Text = "موافق";
OR
string ControlText = myControl.Text;
```

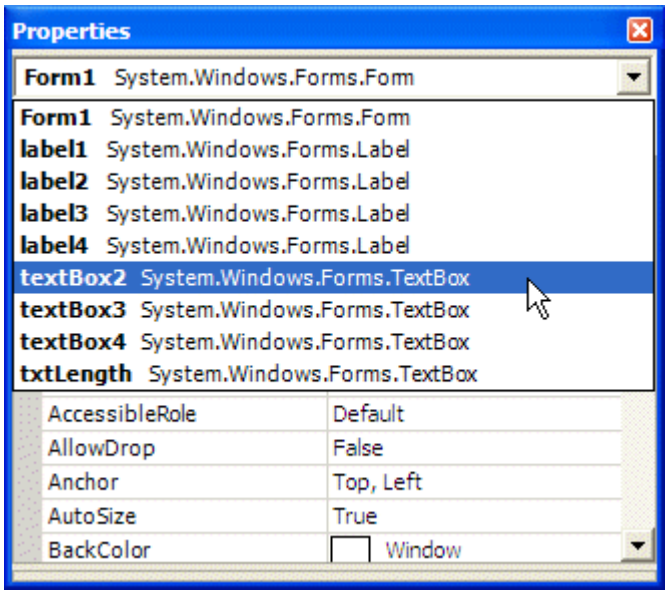
14

Property : Name

وهي بالصيغة التالية :

```
public string Name {get; set;}
```


Senior Developer : Ahmed H. Alawady
Mobil : +2-012-3456808 / Tel : +2-055-3661835 Email : Alawady_ahmed@hotmail.com



الفائدة :

هو إسم الكنترول الذي نستخدمة والذي منه يتم التعامل مع الكنترول مباشرة .
فهو إسم للكلاس الخاص بالكنترول ، فالكنترول TextBox مثلا هو كلاس ولكن كيف نفرق بين أكثر من واحدة يتم إعطاء كل كنترول منها إسم يتم التعامل معه مباشرة .

مثال توضيحي :

من المثال لاسابق الخاص بالخاصية Text ، كنا نتكلم عن زر موافق فيمكن إعطاء إسم للكنترول Button يرمز لهذا الكنترول لكي لا نتوه بين باقي الكنترولز الموجودة ، كما تاه بني إسرائيل ٤٠ سنة في سيناء وبالطبع يمكن أن تقول أخي الحبيب إذا نسمية Ok أنت صحيح ولكن دعنا نقترح إقتراح جميل تخيل إن لديك إثنين من الكنترول أحدهما TextBox والآخر Label بجانبة لتوضيح ماذا سوف يتم كابتة في TextBox مثل هذه الصورة

فهل سوف نكتب إسم TextBox القيمة "Number" وأيضا نكتب للكنترول Label القيمة "Number"

إذا يجب أن نفرق بين إسم كل كنترول بما يناسبة ، فمن هذا نختصر الكنترول كما يلي وبالطبع لك حرية الإختصار

Button بـ btn

TextBox بـ txt

TreeView بـ trv

Label بـ lbl وهكذا .....

إذا من هذا نعود للمثال السابق ونعطي قيمة Button الإسم "btn_Ok" ، والآخر القيمة "btn_Cancel"

طريقة الإستخدام :

بما أنها تقوم بإرسال وإستقبال قيمة (Set & Get)

```
myControl.Name = "ctrl_Name";
OR
string ControlName = myControl.Name;
```

بالنسبة لطلب معرفة إسم الكنترول لإحتياجات كثيرة فمن الممكن طلب إسم الكنترول المرسل تنفيذه عندما يشترك أكثر من كنترول في حدث واحد مثلا ، ونريد أن نعرف من قام بهذا الحدث (أي كنترول) نقوم بالآتي

إذا كان كل من الكنترول btn_Ok والكنترول btn_Cancel مشتركون في نفس الحدث التالي :

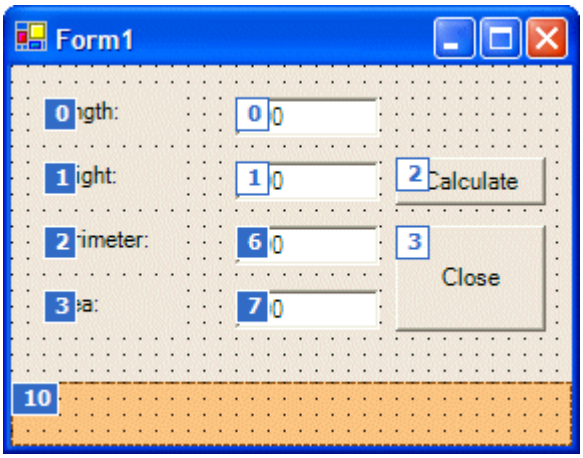
```
private void Control_Click(object sender, System.EventArgs e)
{
if( ((Button)sender).Name == "btn_Ok" )
// Code here
else if( ((Button)sender).Name == "btn_Cancel" )
// Code here
}
```

15

Property : TabIndex

وهي بالصيغة التالية :

```
public int TabIndex {get; set;}
```



الفائدة :

هو ترتيب التنقل والتحرك بين الكنترول في الفورم الواحد .

مثال توضيحي :

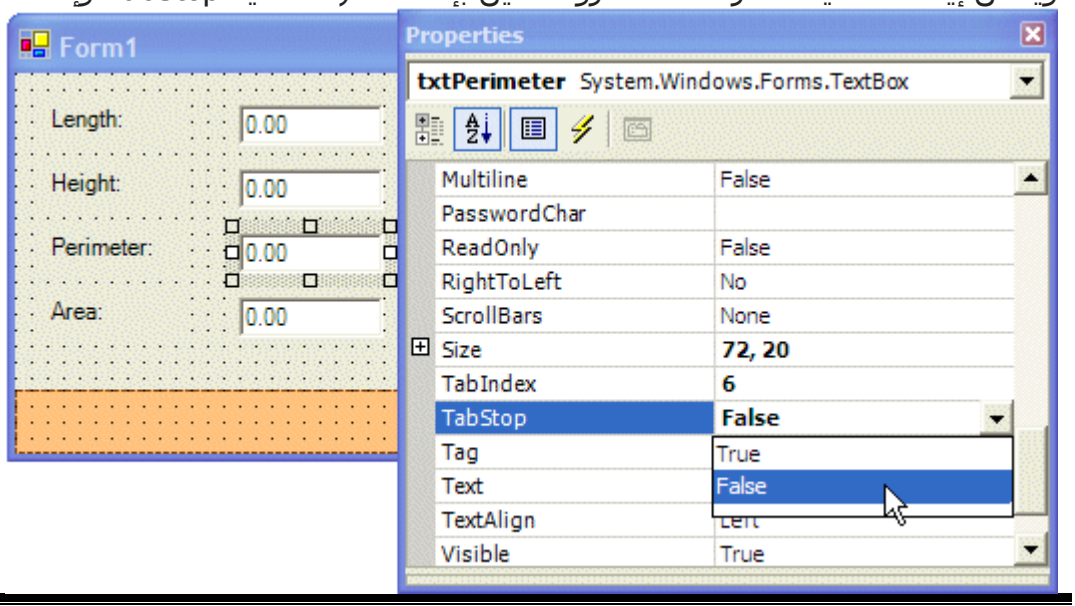
عن الضغط على مفتاح Tab يتم التنقل بين جميع الكنترول الموجودة في الفورم بترتيب الأرقام الذي أعطاه له المبرمج .

طريقة الإستخدام :

بما أنها تقوم بإرسال وإستقبال قيمة (Set & Get)

```
myControl.TabIndex = 1;
OR
int ControlName = myControl.TabIndex;
```

ويمكن إيقاف عملية التحرك عند كنترول معين بإستخدام الخاصية TabStop وإعطاءة قيمة False .



وهنا قد انتهى بحمد الله أشهر الخصائص المشتركة بين جميع الكنترولز ، وسوف أقوم بإذن الله بعمل مثال يشمل جميع ما سبق إن شاء الله ، أسألكم الدعاء

يمكنك متابعة الموضوع فى المنتدى على هذه الوصلة
أو مباشرة من على الرابط : http://www.arabteam2000-forum.com/index.php?showtopic=78785

فهذا هو برنامج يشمل جميع الأوامر السابقة

وصلة البرنامج
أو مباشرة من على : http://www.arabteam2000-forum.com/index.php?act=Attach&type=post&id=26199