



Dotnet France  
Technologies Sharepoint, SQL Server & .NET

Association Dotnet France

# Approche sur les applications Windows Mobile

Quentin Heroguel

# Sommaire

---

1	Introduction.....	3
2	Travailler sur les formulaires .....	3
2.1	Manipulation du formulaire .....	3
2.1.1	Propriété FormBorderStyle .....	3
2.1.2	Propriété WindowState .....	4
2.1.3	Les autres propriétés utiles à la manipulation d'un formulaire .....	4
2.2	Ajouter des composants ergonomiques .....	5
2.2.1	Le clavier virtuel : L'InputPanel .....	5
2.2.2	Le contrôle du menu principal : MainMenu.....	7
2.2.3	Le contrôle ToolBar .....	8
2.2.4	Les ScrollBars .....	9
2.2.5	Le ContextMenu .....	11
3	Les événements.....	12
3.1	Les événements s'apparentant au Tap-and-Hold .....	12
3.2	Les événements des Hardwares Key .....	14
4	Conclusion .....	16

## 1 Introduction

Ce chapitre sera entièrement consacré à la création d'applications pour les plateformes Windows Mobile. Vous découvrirez comment faire une application sous Windows Mobile mais aussi de nouveaux contrôles (par exemple le ToolBar) qui n'ont pas été présentés ou approfondis dans les chapitres précédents. Ainsi que des événements vous permettant de créer d'autres fonctionnalités très utiles pour votre future application.

## 2 Travailler sur les formulaires

Quelque soit le type de projet sous Windows CE ou pour PDA, les classes des formulaires sont identiques. On retrouve dans ces classes les mêmes propriétés, attributs ou événements. Cependant il existe plusieurs contraintes : la taille du formulaire dépend de la taille physique de l'appareil dans lequel l'application sera utilisée, et certaines valeurs de propriétés dépendent aussi de la taille physique de l'application ciblée. De plus, lorsque certaines propriétés ont leur valeur définie, le comportement du formulaire varie selon le type de projet créé (Windows CE ou PDA).

### 2.1 Manipulation du formulaire

Le formulaire comprend plusieurs propriétés utiles pour une meilleure visualisation de votre application ou améliorer l'ergonomie de l'interface utilisateur.

#### 2.1.1 Propriété `FormBorderStyle`

La propriété `FormBorderStyle`, comme en Windows Form, détermine le style de bordures extérieures du formulaire des applications Windows Mobile. Elle affecte aussi l'affichage de la barre de légende ainsi que les boutons mis en place dans cette barre.

Remarque : Elle n'existe pas pour les projets Smartphones et elle comporte une valeur de plus en Windows CE que pour les projets PDA (`FixedDialog`).

Valeur	Description
None	Aucune bordure. Si vous spécifiez cette valeur pour votre formulaire, son redimensionnement ne sera pas effectué correctement.
FixeSingle	Bordure fixe d'une seule ligne (par défaut). Elle permet d'obtenir un redimensionnement correct

	du formulaire.
<code>FixedDialog</code>	Bordure de style boite de dialogue.

L'utilisateur ne peut redimensionner le formulaire, il vous faudra donc définir son redimensionnement dans le code.

### 2.1.2 Propriété `WindowState`

En Windows Mobile, la propriété `WindowState`, qui détermine l'état visuel initial du formulaire, comporte seulement deux paramètres : `WindowState.Normal` (par défaut) et `WindowState.Maximized`. Le `Minimized` n'est pas présent en Windows Mobile quelque soit la nature du projet ( Samrtphone, PDA, Windows CE).

La valeur `Maximized` vous permet de remplir votre écran et ainsi cacher la barre de légende ou encore la barre de menu démarrer mais tout en gardant l'accessibilité du menu principal.

### 2.1.3 Les autres propriétés utiles à la manipulation d'un formulaire

- La propriété `ControlBox` : elle permet d'afficher ou non la boite de contrôle du formulaire (true/false). Pour les Pocket PC il n'existe qu'un seul bouton qui fait office de fermeture ou de réduction de la fenêtre (cela dépend sur quelle valeur est définie la propriété `MinimizedBox`). En Windows CE il y a les trois boutons de contrôle (fermeture, agrandir, réduire).
- La propriété `MinimizedBox` présente pour les Pocket et Windows CE permet d'afficher un bouton qui réduit les fenêtres. Il vous suffit de définir la valeur true pour l'afficher.
- La propriété `MaximizedBox` qui est disponible seulement pour Windows CE, affiche un bouton permettant d'agrandir la fenêtre. Vous devrez mettre comme paramètre true pour qu'elle soit disponible.
- La propriété `Location` définit l'emplacement du formulaire selon des coordonnées (X pour les abscisses, Y pour les ordonnées). `Location` est disponible pour Pocket PC et Windows CE.
- La propriété `Size` définit la taille du formulaire en pixels. Elle est présente pour les types de projet.

## 2.2 Ajouter des composants ergonomiques

### 2.2.1 Le clavier virtuel : L'InputPanel

Le contrôle InputPanel est une adaptation managée du panneau SIP (Soft Input Panel) qui est le clavier virtuel. Il est utilisé pour les Pocket PC et les appareils supportant Windows CE dans le but de saisir des données textuelles (par exemple dans une TextBox). Sur les plateformes Pocket PC, Visual Studio ajoute automatiquement un bouton sur le menu pour activer ou désactiver le panneau SIP. L'implémentation de ce contrôle ne requiert aucune programmation, il vous suffit simplement d'effectuer un Drag-and-Drop de la ToolBox vers le formulaire.

Il est important aussi de bien concevoir votre formulaire lorsque vous implémentez un panneau SIP afin le panneau ne cache pas les autres contrôles. Ces derniers sont redimensionnés selon la taille du panneau SIP implémenté.

Exemple d'activation du panneau SIP quand le contrôle TextBox obtient le focus :

```
//C#
private void textBox1_GotFocus(object sender, EventArgs e)
{
    //Lorsque la TextBox obtient le focus le panneau SIP est activé
    inputPanell.Enabled = true;
}

private void textBox1_LostFocus(object sender, EventArgs e)
{
    //Lorsque la TextBox n'a plus le focus le panneau SIP est désactivé
    inputPanell.Enabled = false;
}
```

```
\VB.NET
Private Sub TextBox1_GotFocus(ByVal sender As System.Object, ByVal
e As System.EventArgs) Handles TextBox1.GotFocus
    'Lorsque la TextBox obtient le focus le panneau SIP est activé
    InputPanell.Enabled = True

End Sub

Private Sub TextBox1_LostFocus(ByVal sender As System.Object, ByVal e
As System.EventArgs) Handles TextBox1.LostFocus
    'Lorsque la TextBox n'a plus le focus le panneau SIP est désactivé
    InputPanell.Enabled = False

End Sub
```



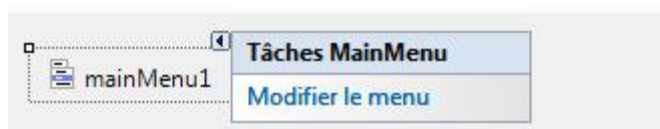
Quelques propriétés peuvent vous être utiles :

Propriété	Description
Bounds	Obtient la taille et l'emplacement du panneau SIP.
Enabled	Active ou désactive le panneau SIP.
VisibleDesktop	Obtient la zone du formulaire qui n'est pas occupée par le panneau SIP.

## 2.2.2 Le contrôle du menu principal : MainMenu

Le MainMenu est un contrôle qui intègre un menu dans votre formulaire. Chaque formulaire de votre projet pour Pocket PC ajoute automatiquement un contrôle MainMenu. Pour les projets Windows CE ces contrôles ne sont pas ajoutés automatiquement.

Le MainMenu est composé d'éléments que vous ajoutez, appelés MenuItem. Pour ajouter des éléments il vous suffira de sélectionner votre contrôle qui se situe en bas du designer de votre formulaire et de cliquer sur la flèche qui s'affiche lorsque vous avez le focus du contrôle. Ensuite sélectionnez "Modifier le Menu".



La propriété RightToLeft vous permettra d'afficher le texte de droite à gauche pour certaines langues qui l'exigent (l'arabe par exemple). Vous avez aussi la méthode CloneMenu vous permettant de copier un menu déjà conçu.

### 2.2.3 Le contrôle ToolBar

Le contrôle ToolBar permet d'afficher des boutons fréquemment utilisés appelés ToolBarButton (ce contrôle est inexistant pour les projets Smartphones). La seule différence entre le ToolBar d'un projet Pocket PC et celui d'un projet Windows CE est sa disposition : le contrôle est affiché en bas pour les Pocket PC et il est affiché en haut pour les applications Windows CE.

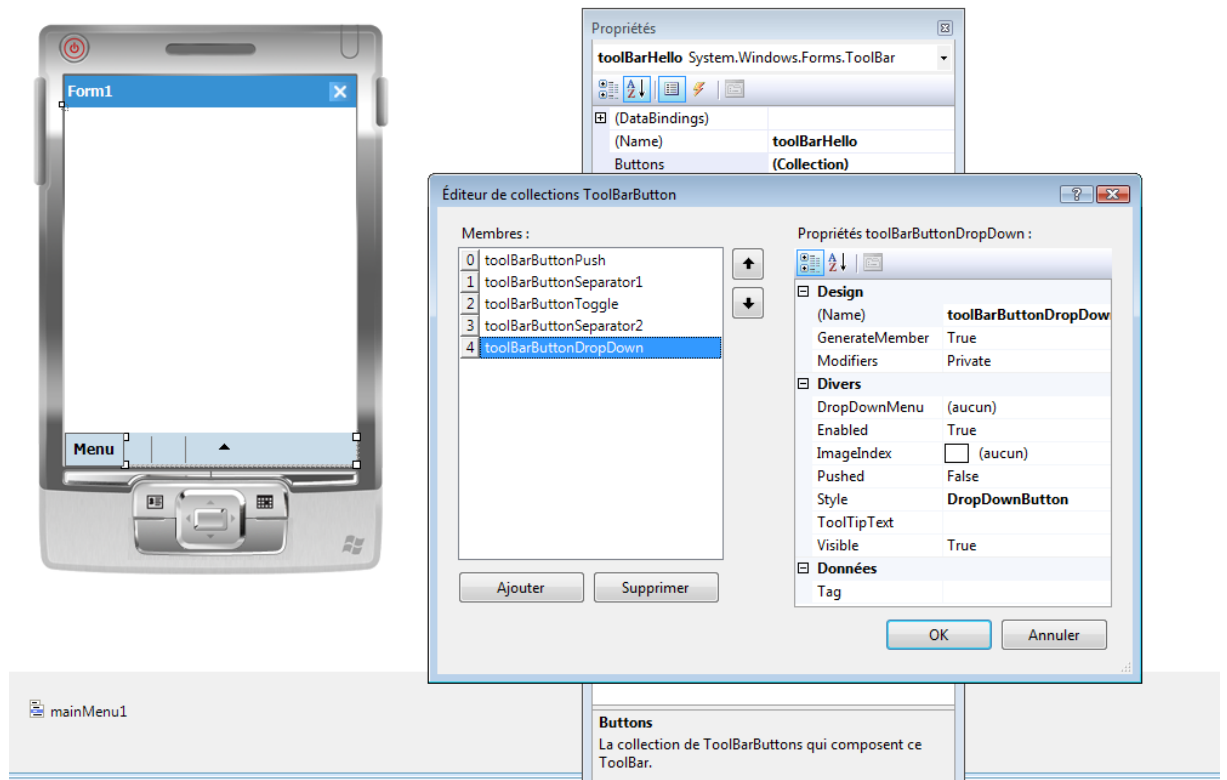
La propriété Buttons vous permettra d'ajouter des ToolBarButton à votre contrôle. Depuis la partie design lorsque vous accédez à la propriété Buttons, vous ouvrez une nouvelle fenêtre qui n'est autre que l'éditeur de collections des ToolBarButton.

D'autres propriétés vous permettront d'améliorer votre contrôle ToolBar comme ImageList qui ajoute des images à vos boutons ou encore la propriété Appearance qui définit l'apparence du ToolBar (par exemple en 2D).

De plus les propriétés des ToolBarButton vous permettront aussi de les personnaliser :

Propriété	Description
DropDownMenu	Définit le menu à afficher dans un bouton déroulant de la barre d'outils (ContextMenu). Vous devrez donc spécifier dans la propriété Style DropDownButton pour ainsi affecter un ContextMenu à un ToolBarButton.
Enabled	Définit si le bouton est activé ou non.
ImageIndex	Indique la valeur d'index de l'image assignée au bouton de la propriété ImageList du ToolBar.
Pushed	Lorsque la propriété Style est définie sur ToggleButton, si Pushed est activé (true) alors le bouton apparaît enfoncé par rapport aux autres boutons (appelé bouton à bascule).
Style	Détermine le style du bouton. Le style PushButton est un bouton standard en 3D, le style ToggleButton est un bouton à bascule, le style Separator permet d'afficher un séparateur et enfin le style DropDownButton permet d'obtenir un bouton déroulant un menu d'un contrôle ContextMenu.
Tag	Détermine l'objet qui contient les données du bouton.

ToolTipText	Affiche une info-bulle lorsque le bouton a le focus.
Visible	Définie si le bouton est visible ou non.



Voici un aperçu d'ajouts de différents boutons de la ToolBar. Dans l'ordre de gauche à droite sur l'émulateur (ou bien de haut en bas dans l'éditeur de collections ToolBarButton): PushButton, Separator, ToggleButton, Separator, DropDownButton.

## 2.2.4 Les ScrollBars

Lorsque la propriété AutoScroll n'est pas activée, vous avez la possibilité d'ajouter des barres de défilements à votre formulaire (Scroll bar). En Windows Mobile il existe deux types de barres de défilement : la verticale (VScrollBar) et l'horizontale (HScrollBar) qui sont deux contrôles distincts dans la boîte à outils.

Les propriétés les plus utiles sont au nombre de quatre :

Propriété	Description
Dock	Définit l'emplacement du contrôle qui permet d'ancrer le contrôle à la bordure choisie du conteneur (du formulaire).
LargeChange	Définit la valeur de la case de défilement par rapport à de longues distances.
SmallChange	Définit la valeur de la case de défilement par rapport à de courtes distances.
Value	Définit la position de la case de défilement selon une valeur numérique.

L'exemple ci-dessous vous démontre comment ajouter un contrôle Scroll depuis votre code :

```
//C#
public Form1 ()
{
    InitializeComponent();
    //On crée un nouveau contrôle ScrollBar
    HScrollBar hScrollBar1 = new HScrollBar();
    //On ancre le contrôle en bas du formulaire
    hScrollBar1.Dock = DockStyle.Bottom;
    hScrollBar1.LargeChange = 30;
    hScrollBar1.Value = 20;
    hScrollBar1.SmallChange = 1;
    //On ajoute le contrôle au formulaire
    Controls.Add(hScrollBar1);
}
```

```
\VB.NET
Private Sub InitializeComponent ()

    'On crée un nouveau contrôle ScrollBar
    Dim hscrollbar1 As HScrollBar = New HScrollBar
    'On ancre le contrôle en bas du formulaire
    hscrollbar1.Dock = DockStyle.Bottom
    hscrollbar1.LargeChange = 30
    hscrollbar1.SmallChange = 1
    hscrollbar1.Value = 20
    'On ajoute le contrôle au formulaire
    Controls.Add(hscrollbar1)
End Sub
```

Remarque : N'oubliez pas de désactiver la propriété AutoScroll de votre formulaire si vous voulez ajouter ces contrôles et les configurer à votre guise.

## 2.2.5 Le ContextMenu

Le contrôle ContextMenu est un contrôle pop-up lié à un autre contrôle ou à un formulaire qui affiche un menu. Ce contrôle est le même qui s'affiche lorsque vous faites un clic droit sur votre bureau par exemple avec les options d'affichage, de triage et autres. Cependant en Windows Mobile, le contrôle ContextMenu ne s'active pas en faisant un clic droit pour les Pocket PC mais en maintenant le stylet (ou en faisant un clic gauche de votre souris si vous utilisez un émulateur) sur l'écran pendant environ deux secondes : le tap-and-hold (littéralement "appuyez-et-maintenez"). Pour Windows CE il vous suffira simplement de faire un clic droit comme vous avez l'habitude.



Pour ajouter un contrôle ContextMenu depuis le designer, il vous suffira de glisser le contrôle de votre boîte à outils vers le formulaire. Le contrôle s'affiche comme un composant du formulaire en bas du designer. Ensuite il vous reste à ajouter les éléments composant votre contrôle comme vous le feriez avec un MainMenu.



**N**'oubliez pas d'assigner à votre contrôle ou à votre formulaire le ContextMenu que vous voulez lui attribuer. Pour cela vous devrez choisir le nom de votre ContextMenu dans la propriété ContextMenu de votre formulaire ou de votre contrôle.

## 3 Les événements

### 3.1 Les événements s'apparentant au Tap-and-Hold


Dans le cas où vous voudriez afficher une autre fonction qu'un ContextMenu possédant le même principe d'affichage. Il vous suffit de gérer les événements Mouse\_Down, Mouse\_Up ou encore Lost\_Focus. Voici la liste des événements permettant d'ajouter d'autres fonctions à la manière d'un ContextMenu :

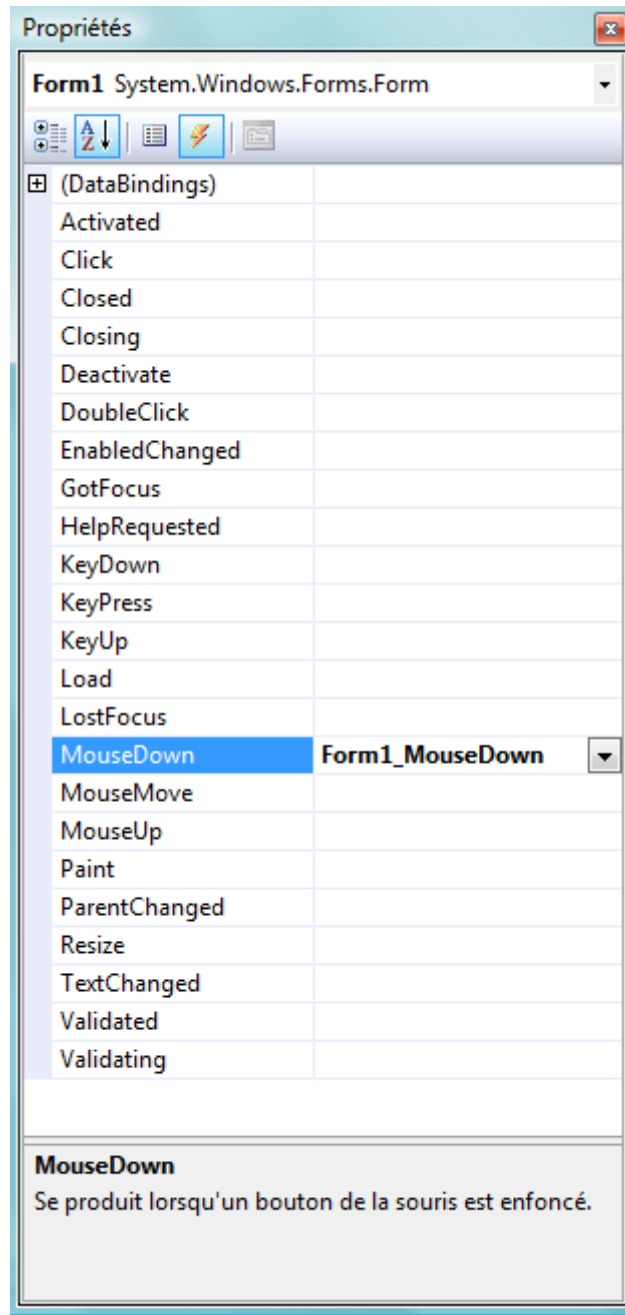
Événement	Description
Get_Focus	Déclenche l'événement lorsque le contrôle possède le focus.
Lost_Focus	Déclenche l'événement lorsque le contrôle perd le focus.
Mouse_Down	Déclenche l'événement lorsqu'un bouton de la souris est enfoncé sur le contrôle.
Mouse_Move	Déclenche l'événement lorsque le curseur de la souris se déplace sur le contrôle.
Mouse_Up	Déclenche l'événement lorsqu'un bouton de la souris est relâché sur le contrôle

Voici un exemple très simple de l'utilisation d'un de ces événements, lorsque vous enfoncerez un bouton de la souris n'importe où dans le formulaire, une MessageBox s'affichera :

```
//C#
private void Form1_MouseDown(object sender, MouseEventArgs e)
{
    MessageBox.Show("Hello");
}
```

```
\VB.NET
Private Sub Form1_MouseDown(ByVal sender As System.Object, ByVal e As
System.Windows.Forms.MouseEventArgs) Handles MyBase.MouseDown
    MessageBox.Show("Hello");
End Sub
```

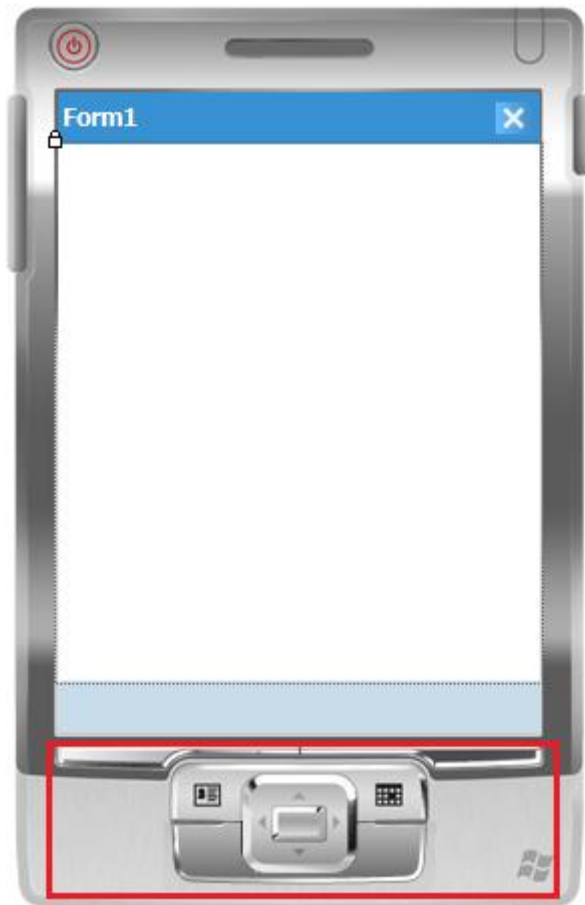
Pour créer votre événement, il vous suffit d'accéder aux propriétés de votre contrôle depuis la partie design de votre projet. Sélectionnez l'icône  qui vous affichera la liste de tous les événements disponibles pour le contrôle. Ensuite double cliquez sur un nom d'événement comme sur le screenshot suivant :



Visual Basic créera automatiquement l'événement et vous amènera dans le code behind de votre projet. Vous pourrez ainsi intégrer les fonctions que vous voulez attribuer à votre événement.

## 3.2 Les événements des Hardwares Key

Les Hardwares Key sont les boutons physiques du pocket PC.



Pour accéder à l'événement de chacun d'entre eux il vous suffit de sélectionner l'événement `Key_Down` depuis les propriétés de votre formulaire. En double-cliquant sur l'événement, vous accédez au code behind. Une condition est créée pour chaque bouton Key, il vous suffit donc d'ajouter les fonctionnalités que vous voulez leur attribuer comme dans l'exemple suivant :



```
//C#
private void Form1_KeyDown(object sender, KeyEventArgs e)
{
    if ((e.KeyCode == System.Windows.Forms.Keys.Up))
    {
        MessageBox.Show("Up");
    }
    if ((e.KeyCode == System.Windows.Forms.Keys.Down))
    {
        MessageBox.Show("Down");
    }
    if ((e.KeyCode == System.Windows.Forms.Keys.Left))
    {
        MessageBox.Show("Left");
    }
    if ((e.KeyCode == System.Windows.Forms.Keys.Right))
    {
        MessageBox.Show("Right");
    }
    if ((e.KeyCode == System.Windows.Forms.Keys.Enter))
    {
        MessageBox.Show("Enter");
    }
}
```

```
'VB.NET
Private Sub Form1_KeyDown(ByVal sender As System.Object, ByVal e As
System.Windows.Forms.KeyEventArgs) Handles MyBase.KeyDown
    If (e.KeyCode = System.Windows.Forms.Keys.Up) Then

        MessageBox.Show("Up")
    End If
    If (e.KeyCode = System.Windows.Forms.Keys.Down) Then

        MessageBox.Show("Down")
    End If
    If (e.KeyCode = System.Windows.Forms.Keys.Left) Then

        MessageBox.Show("Left")
    End If
    If (e.KeyCode = System.Windows.Forms.Keys.Right) Then

        MessageBox.Show("Right")
    End If
    If (e.KeyCode = System.Windows.Forms.Keys.Enter) Then

        MessageBox.Show("Enter")
    End If

End Sub
```

## 4 Conclusion

Dans ce chapitre vous avez donc pu voir les différentes fonctionnalités pour vous permettre de personnaliser, d'améliorer et d'approfondir le fonctionnement de votre formulaire Windows Mobile. Les contrôles et les événements énumérés dans ce chapitre permettent à l'utilisateur d'obtenir une meilleure facilité d'utilisation de votre application.