

Les nouveautés de Delphi 2009

par Bruno Guérangé (nono40.developpez.com)

Date de publication : 24 septembre 2008

Dernière mise à jour : 29 septembre 2008

Cet article présente les nouveautés du nouvel IDE de Codegear : Delphi 2009.

I - Introduction.....	3
II - Unicode.....	5
II-A - Unicode dans le langage.....	5
II-B - Unicode dans la VCL.....	6
III - Classes génériques.....	6
IV - Autres nouveautés du langage.....	7
IV-A - TStringBuilder.....	7
IV-B - Evolution de TObject.....	7
IV-C - Méthodes anonymes.....	7
IV-D - Exit accepte un paramètre facultatif.....	8
V - Gestionnaire de ressources.....	9
VI - Explorateur de classes.....	9
VII - Nouveautés de la VCL.....	12
VII-A - Nouveaux composants.....	12
VII-A-1 - TCategoryPanelGroup.....	12
VII-A-2 - TButtonedEdit.....	12
VII-A-3 - TLinkLabel.....	13
VII-A-4 - TBallonHint.....	13
VII-A-5 - TRibbon.....	14
VII-B - Evolution des composants de base.....	14
VII-C - Support du PNG.....	15
VII-D - TImageList accepte tous les types d'image.....	15
VIII - Informations et téléchargement.....	15

I - Introduction

Delphi 2009 connu sous le nom de code Tiburon est maintenant disponible.

De nombreuses nouveautés sont incluses dans cette version. Certaines sont des avancées majeures, d'autres ne sont que des évolutions simples.

Nous allons ici vous présenter les nouveautés principales.

Soft Windows - Unit1

projet Exécuter Composant Outils Fenêtre Aide Disposition par défaut



Page Bienvenue Unit1

```

50 Cells[0,1] := 'Espagnol';
    Cells[1,1] := 'Fecha de inicio';
    Cells[2,1] := 'Fecha de fin';
    Cells[0,2] := 'Anglais';
    Cells[1,2] := 'Start date';
    Cells[2,2] := 'End date';
    Cells[0,3] := 'Chinois simplifié';
    Cells[1,3] := '开始日期';
    Cells[2,3] := '结束日期';
    Cells[0,4] := 'Turc';
60 Cells[1,4] := 'Başlangıç tarihi';
    Cells[2,4] := 'Bitiş tarihi';
    Cells[0,5] := 'Chinois traditionnel';
    Cells[1,5] := '開始日期';
    Cells[2,5] := '結束日期';

    End;
    ListBox1.ItemIndex := 3;
end;

70 procedure TForm1.計劃Click(Sender: TObject);
    Var 來源:Integer;
71 begin
    來源 := 1234.
    
```



71: 20

Insertion

Modifié

Code

Conception

Historique

II - Unicode

C'est la principale évolution de Delphi 2009 par rapport aux versions précédentes. L'utilisation de l'Unicode est maintenant complètement intégré dans le langage, la VCL et l'IDE.

Nombreux sont ceux qui ont eu à gérer des encodages plus ou moins difficiles ou utiliser des composants tiers pour gérer l'affichage de langues non latines comme l'Arabe ou le Chinois. Avec cette nouvelle version de Delphi, tout est Unicode.

II-A - Unicode dans le langage

C'est le grand changement de l'intégration de l'Unicode : le type **String** est maintenant équivalent au type **UnicodeString** (de même que **Char** et **PChar** sont équivalents à **WideChar** et **PWideChar**). Ce changement permet de stocker des chaînes Unicode dans n'importe quelle propriété d'un composant sans devoir réécrire le code.

```

• procedure TForm1.Button1Click(Sender: TObject);
•   var Chaîne:String;
•   begin
•     Chaîne := '你確定使用如下指令嗎?';
30   ShowMessage('清潔');
•   end;
    
```

Attention tout de même: ce changement demandera à vérifier/modifier une partie du code existant, surtout si vous faites appel aux **PChar**. Le type **Char** étant maintenant sur deux octets au lieu d'un, tous les opérations faites avec des **Char** et **PChar** pour traiter des octets deviendront fausses. De nombreux warnings sont générés par le compilateur dans ce cas, ils vous aideront à mettre à jour le code. Notez que si votre code doit rester en codage Ansi dans Delphi 2009, il suffit de déclarer les variables en **AnsiChar** ou lieu de **Char**, **AnsiString** ou lieu de **String** et **PAnsiChar** ou lieu de **PChar**.

L'évolution du langage permet aussi d'utiliser des caractères unicode dans le code pour le nommage des identificateurs, que ce soit pour des variables, des composants ou même des Unités.

```

- Function Carré(Valeur:Integer):Integer;
•   Begin
•     Result := Valeur*Valeur;
•   End;
•
90 procedure TForm1.計劃Click(Sender: TObject);
•   Var 來源:Integer;
•   begin
•     來源:=1234;
•     來源:=來源+2;
•     ShowMessage(IntToStr(Carré(來源)));
•   end;
    
```



Si votre code doit être aussi utilisé par les versions précédentes de Delphi, il est recommandé de n'utiliser que les caractères habituels pour les noms de variables.

i Pour en connaître davantage sur l'intégration de l'unicode dans Delphi 2009, consultez l'article de Damien Griessinger
Présentation : Delphi 2009 et l'UNICODE

II-B - Unicode dans la VCL

La VCL a été revue pour utiliser de l'unicode dans toutes les propriétés contenant des chaînes :

Français	Date de début	Date de fin
Espagnol	Fecha de inicio	Fecha de fin
Anglais	Start date	End date
Chinois simplifié	开始日期	结束日期
Turc	Başlangıç tarihi	Bitiş tarihi
Chinois traditionnel	開始日期	結束日期

III - Classes génériques

Les classes génériques sont la deuxième nouveauté importante dans Delphi 2009. Elles existaient déjà avec la version .NET de Delphi, elles sont maintenant aussi disponibles pour Delphi Win 32.

Une classe générique est un modèle de classe définissant des comportements communs sur des types différents. Exemple issu des sources de Delphi :

```

type
  TList<T> = class(TEnumerable<T>)
    // ...
  public
    // ...


    function Add(const Value: T): Integer;
    procedure Insert(Index: Integer; const Value: T);

    function Remove(const Value: T): Integer;
    procedure Delete(Index: Integer);
    procedure DeleteRange(AIndex, ACount: Integer);
    function Extract(const Value: T): T;

    procedure Clear;

    property Count: Integer read FCount write SetCount;
    property Items[Index: Integer]: T read GetItem write SetItem; default;
  end;
    
```

Cette Classe **TList<T>** définit le comportement d'une liste d'objets T. Ensuite, si vous avez besoin d'une liste d'entiers, vous utiliserez la classe **TList<Integer>**.

Si vous désirez en connaître plus, vous pouvez consulter l'article de Laurent Dardenne :  [Les génériques sous Delphi .NET](#)

IV - Autres nouveautés du langage

IV-A - TStringBuilder

La classe **TStringBuilder** est une nouvelle classe de construction de chaîne. Elle est plus rapide que le traitement classique du String, car elle contient un buffer interne qui n'est pas alloué/désalloué à chaque modification. Elle ne contient pas non plus de compteur de référence.

Cette classe s'utilise comme la classe System.Text.StringBuilder du framework Dotnet, facilitant ainsi la compatibilité d'application entre Delphi pour Win32 et Delphi pour .NET.

IV-B - Evolution de TObject

Le type **TObjet** comporte trois nouvelles méthodes virtuelles :

- **ToString** : retourne une chaîne représentant l'objet.
- **GetHashCode** : retourne un code d'identification de l'objet, ce code permet de comparer le contenu de deux objets différents de la même classe.
- **Equals** : compare l'instance de l'objet avec l'instance passée en paramètre

Ces méthodes étant virtuelles, il appartient aux classes descendantes de les implémenter. Le fait qu'elles soient définies au plus bas niveau permet de les utiliser sur n'importe quelle classe d'objet. A noter que ces méthodes existent déjà sur le **TObject** de base du framework .NET.

IV-C - Méthodes anonymes

Une méthode est une méthode sans nom dont le code lui-même est stocké dans une variable. Le code affecté garde une trace de l'environnement et des variables locales utile à son fonctionnement. Il n'est pas exécuté à l'affectation de la variable qui le contient, mais quand cette variable est elle-même appelée.

Pour utiliser les méthodes anonymes il faut toujours créer un type définissant le prototype d'appel de la méthode anonyme. Ici c'est une fonction retournant un entier.

```
Type
TCompteur = Reference To Function:Integer;
```

Une variable de ce type est alors affectée en recevant directement le code *inline* ou par affectation d'une autre fonction anonyme du même type.

Ici sur le bouton BtnCreer, on affecte le code devant être utilisé :

```
type
  TForm4 = class(TForm)
    ...
  private
    { Déclarations privées }
    Compte : TCompteur;
```

```

    ...
end;

//.....

procedure TForm4.BtnCreerClick(Sender: TObject);
Var i:Integer;
begin
    // Initialisation variable locale
    i:=0;
    // On affecte la fonction anonyme à Compte qui
    // se chargera d'en garder la référence
    Compte := Function:Integer
    Begin
        // I est utilisé dans le code bien qu'étant une variable
        // locale de BtnCreerClick()
        Result := i;
        Inc(i);
        if i>2 then
            i:=0;
    End;
end;

```

Une fois que l'on a cliqué sur le bouton BtnCreer, on peut utiliser la variable Compte pour générer la valeur suivante du compteur. Par exemple on l'affiche une dizaine de fois :

```

procedure TForm4.BtnUtiliseClick(Sender: TObject);
Var N:Integer;
begin
    // On utilise Compte, le compteur va bien suivre la séquence
    // 0 1 2 0 etc...
    for N := 0 to 9 do
        Mem1.Lines.Add(IntToStr(Compte));
    end;

```

IV-D - Exit accepte un paramètre facultatif

La procédure **Exit** accepte maintenant un paramètre facultatif du même type que le type de retour de la fonction ou il est placé. Ce paramètre est la valeur de retour de la fonction quand l'**Exit** est exécuté.

Par exemple :

```

Function Test(Valeur:Integer):Integer;
Begin
    // test valeur hors limite
    if Valeur<0 then
        Exit(-1);
    // Traitement
    Result:=Round(SQRT(Valeur));
End;

procedure TForm1.Button2Click(Sender: TObject);
begin
    ShowMessage(IntToStr(Test(-20))); // Affiche -1
    ShowMessage(IntToStr(Test(64))); // Affiche 8
end;

```

Cette nouveauté permet d'alléger le code, dans Delphi 2007 il faudrait écrire :

```

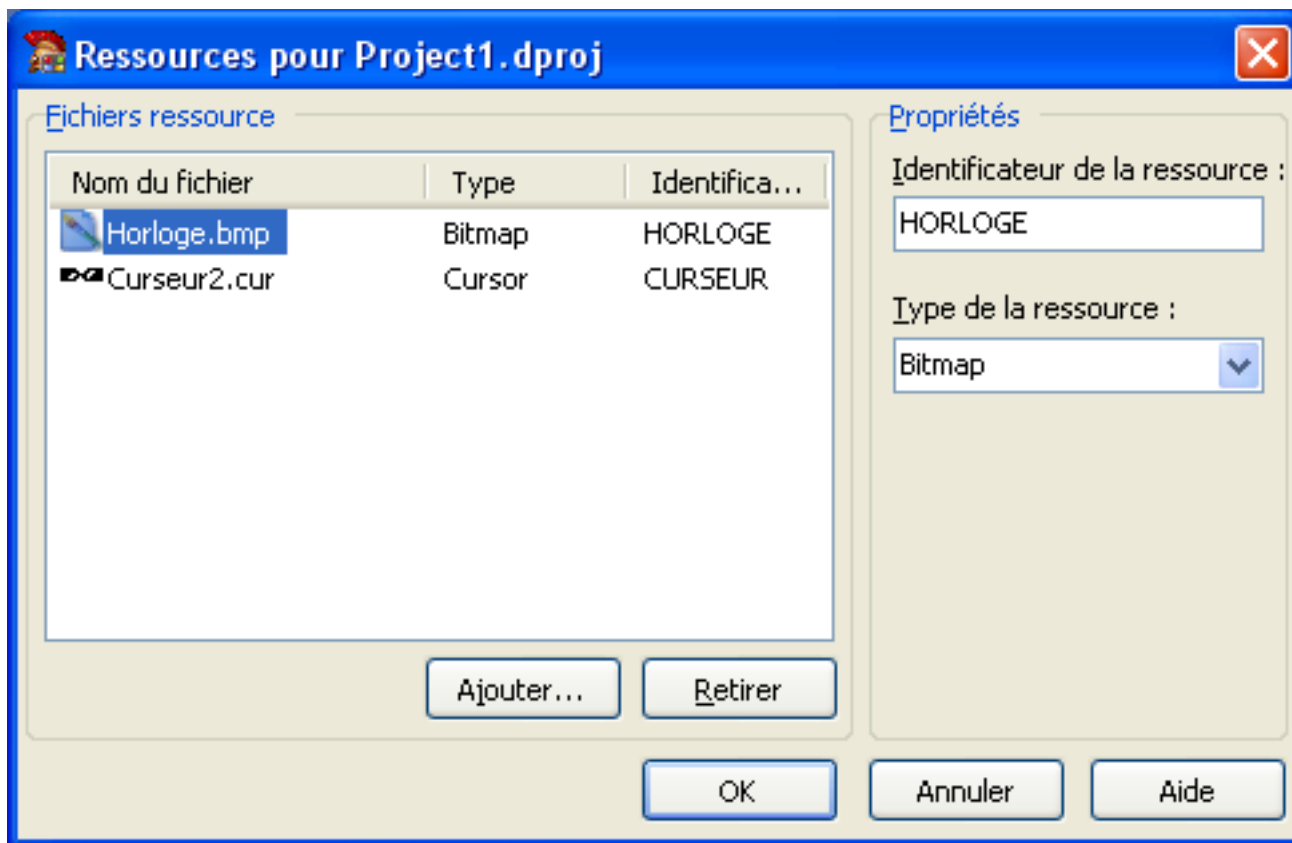
// test valeur hors limite
if Valeur<0 then
begin
    Result := -1;
    Exit;

```

end;

V - Gestionnaire de ressources

L'ajout, dans Delphi 2009, d'un gestionnaire de ressources facilite grandement leur gestion dans un projet. Cet utilitaire est disponible dans le menu *Projet->Ressources* de l'IDE. Dans l'exemple suivant, nous avons ajouté une image et un curseur :



Plus besoin de créer un .RES ni d'utiliser la directive \$R. Les ressources sont incluses dans la compilation automatiquement.

Toujours avec l'exemple ci-dessus on peut utiliser le curseur comme suit :

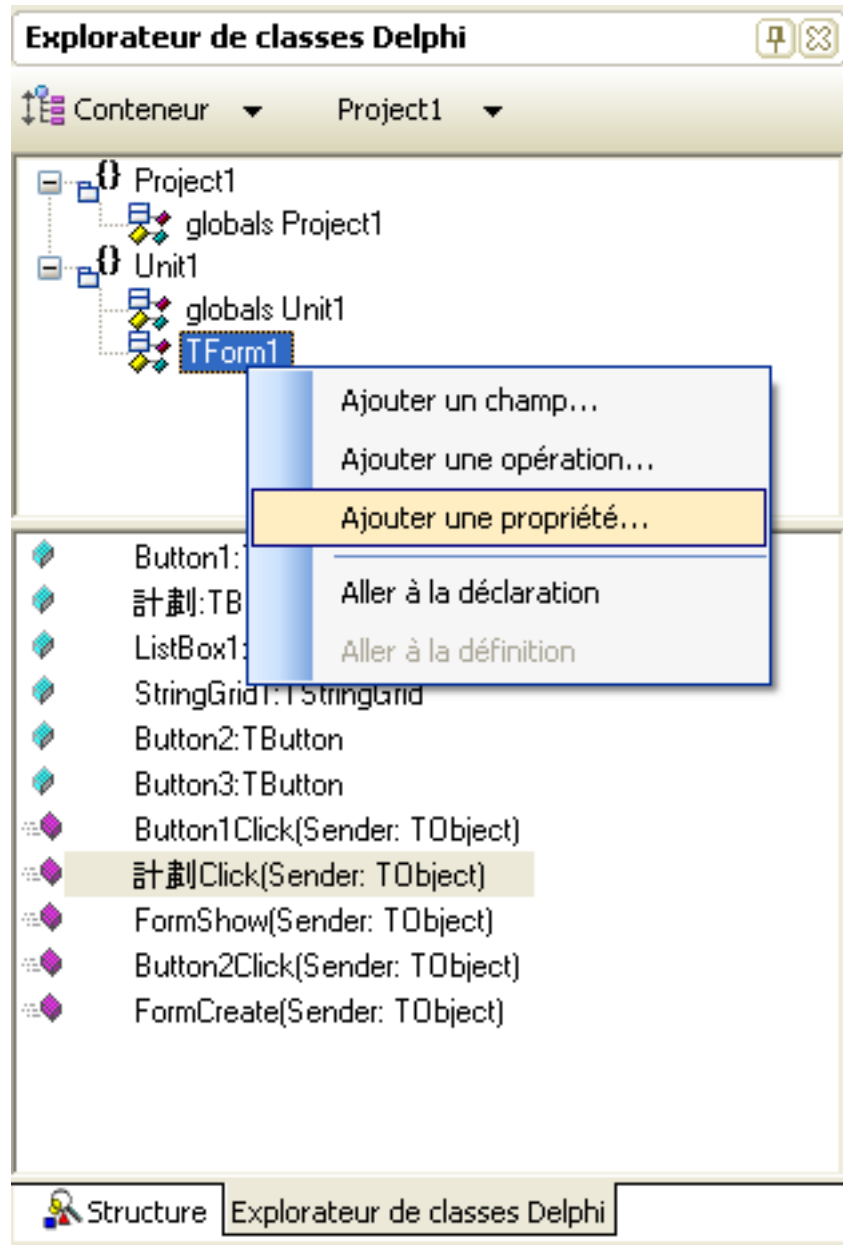
```

Const
  crCurseur1=1;

procedure TForm1.FormCreate(Sender: TObject);
begin
  // Le curseur est alors lu dans les ressources et ajouté à la liste des
  // curseurs utilisables par l'application
  Screen.Cursors[crCurseur1] := LoadCursor(HInstance, 'CURSEUR');
  // Le curseur est utilisable comme ceux prédéfinis :
  Button1.Cursor := crCurseur1;
end;
  
```

VI - Explorateur de classes

Un nouvel explorateur de classes permet de parcourir l'ensemble des classes de votre projet. Pour l'afficher, il faut aller dans le menu *Voir->Explorateur de classes*, il se positionne dans la même fenêtre que l'explorateur de structure de l'unité :



A partir de cet explorateur, il est possible de se positionner rapidement dans le code mais aussi d'ajouter des champs ou propriétés.

Par exemple, nous créons la classe vide suivante :

```
Type
TTest=Class (TObject)
End;
```

Dans l'explorateur de classes, TTest est bien ajouté à la liste. Si on effectue un click droit sur cette classe et ensuite *Ajouter une propriété*, une fenêtre permet de renseigner les paramètres :

Nous avons juste donné le nom (NombreTest) et coché la case *Créer un champ*, le reste est rempli automatiquement avec les normes d'usage.

En cliquant sur OK le code devient :

```

Type
TTest=Class(TObject)
strict private
    function GetNombre : Integer;
    procedure SetNombre(val : Integer);
public
    property Nombre : Integer read GetNombre write SetNombre;
strict private
var
    FNombre:Integer;
End;

//.....//

function TTest.GetNombre: Integer;
begin
    result := FNombre;
end;

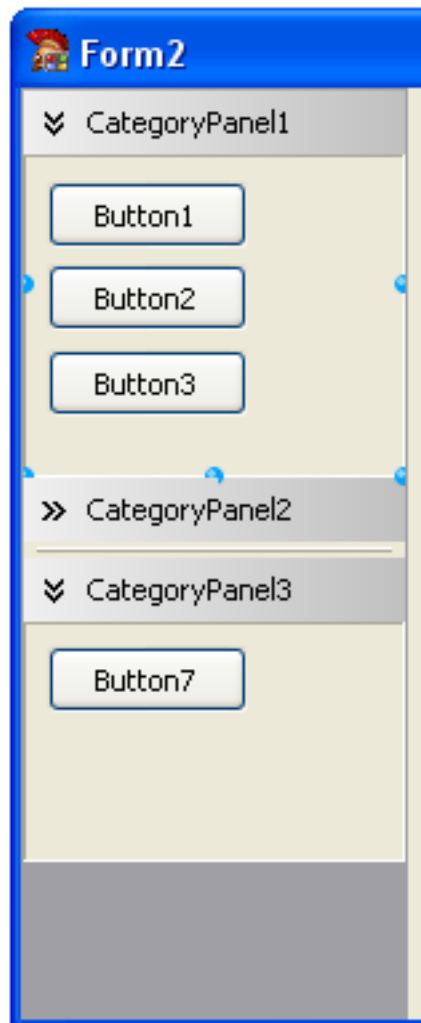
procedure TTest.SetNombre(val : Integer);
begin
    FNombre := val;
end;
    
```

VII - Nouveautés de la VCL

VII-A - Nouveaux composants

VII-A-1 - TCategoryPanelGroup

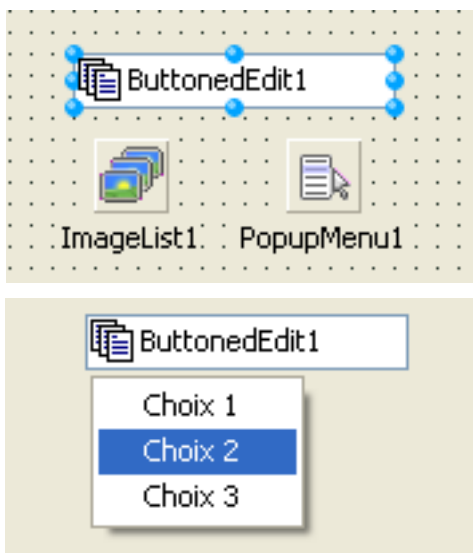
Le **TCategoryPanelGroup** permet de réaliser rapidement une palette d'outils ou d'options ressemblant à celle des composants de l'éditeur de fiches. Ce composant contient un ensemble de **TCategoryPanel** repliables.



C'est ici sa présentation pas défaut mais il est possible de changer complètement l'apparence via l'inspecteur d'objets.

VII-A-2 - TButtonEdit

Ce composant est un dérivé de **TEdit** sur lequel il est possible d'ajouter un bouton sur la gauche et un sur la droite. Ces boutons sont dessinés via un **TImageList**. L'appui sur l'un des boutons génère un évènement ou plus simplement ils peuvent être associés à un **TPopupMenu** pour faire une liste déroulante.



VII-A-3 - TLinkLabel

Ce composant permet de mettre un label contenant un lien hypertexte cliquable. Il n'y a pas de propriété particulière: le champ Caption doit contenir un lien (ou des liens) composés à l'aide de la balise HTML <a>.

```
LinkLabel1.Caption := 'C'est un lien vers <a href="http://www.developpez.com">www.developpez.com</a>';
```


Les liens sont bien dessinés avec une autre couleur. un évènement particulier est déclenché sur le click sur un lien, cet évènement contient en paramètre l'adresse du lien.

C'est un lien vers www.developpez.com

VII-A-4 - TBallonHint

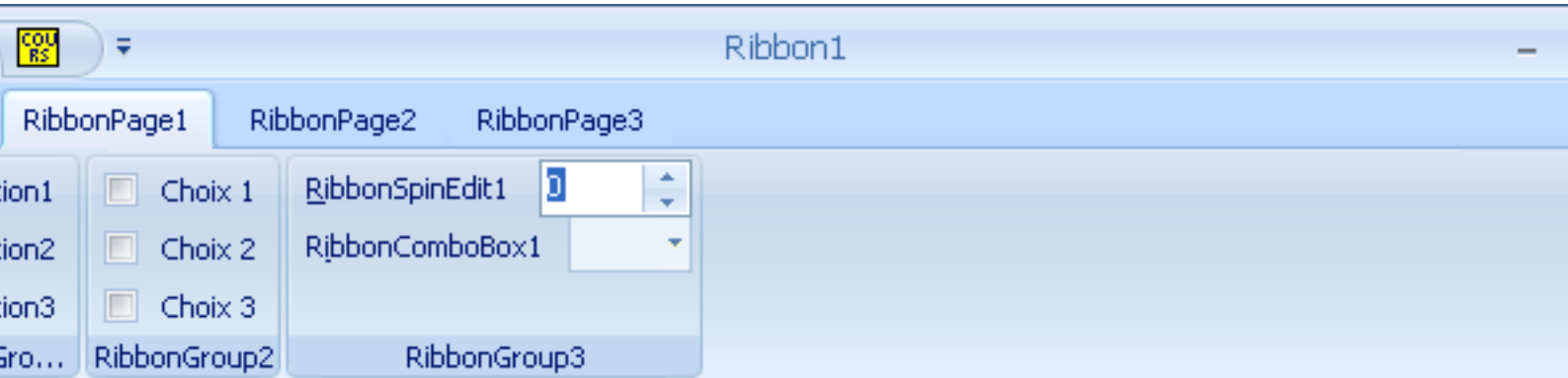
Ce composant est un composant non visuel permettant de modifier l'apparence du hint affiché sur un composant. Une fois placé sur la fiche il faut l'associer aux composants concernés par la nouvelle propriété **CustomHint**.



 A noter qu'il est possible aussi de définir tous les composants d'une fiche en associant le **BallonHint** directement à la fiche et en laissant la nouvelle propriété **ParentCustomHint** à **True** pour les composants contenus.

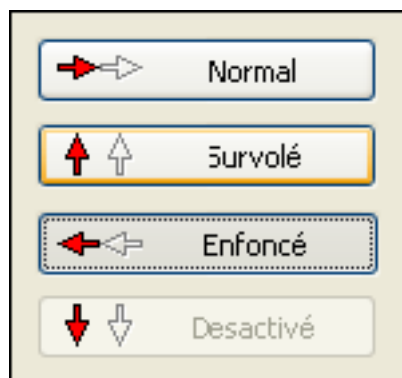
VII-A-5 - TRibbon

Ce composant permet de donner à vos application des barre d'outils semblables à celles d'office 2007. Voici en quelques clics de souris ce que l'on peut obtenir :

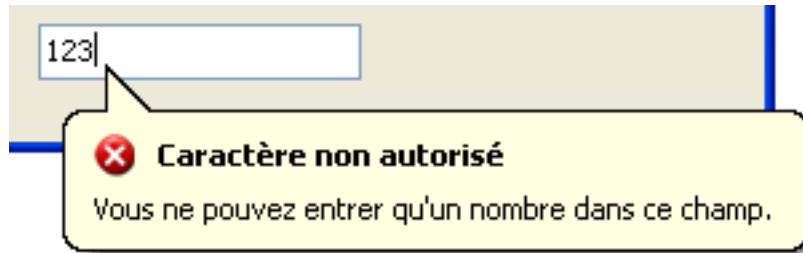


VII-B - Evolution des composants de base

Delphi 2009 permet maintenant d'intégrer des images dans le TButton de base, ces images sont personnalisables pour les modes normal, survolé, enfoncé et désactivé du bouton. L'image peut être centrée afin de remplacer le caption du bouton.



Le TEdit contient une propriété **NumbersOnly** permettant de n'entrer que des chiffres. Quand un caractère non autorisé est entré, une bulle s'affiche pour le signaler.



VII-C - Support du PNG

Le **TImage** supporte maintenant le PNG, y compris les effets de transparence.

Dans l'image ci-dessous le logo CodeGear au format PNG est dans un **TImage**, cette image est placée sur sur trois **TShape** bleu, blanc et rouge afin de faire ressortir les ombrages.



VII-D - TImageList accepte tous les types d'image

Le **TImageList** permet maintenant de contenir tous les formats lisibles dans un **TImage**. Il n'est plus nécessaire de tout convertir en BMP pour remplir le **TImageList**.

VIII - Informations et téléchargement.

Si vous souhaitez télécharger la version entièrement fonctionnelle pendant 14 jours, vous pouvez vous rendre sur le site de téléchargement de CodeGear :

Télécharger la trial de delphi 2009

Bien que la page de téléchargement soit en Anglais, Delphi 2009 Trial est bien en Français.

En complément d'information, vous pouvez consulter la fiche technique en Français de Delphi 2009 :

CodeGear Delphi 2009 Fiche Technique FR

Comparez les caractéristiques des différentes versions Architecte, Entreprise et Professionnelle afin de mieux guider votre choix sur la version qui vous convient :

CodeGear Delphi 2009 matrice des fonctionnalités

En cas de question sur la nouvelle version de Delphi, vous pouvez consulter la FAQ Delphi 2009 de CodeGear :

CodeGear Delphi 2009 F.A.Q.