Cours Delphi de BOUNCEUR Ahcène

Table des matières

	Page
Chapitre 1	
Introduction	**6**
Que ce que Delphi	6
Utilisation du modèle objet	6
Que ce qu'un objet	6
Chapitre 2	
Environnement Delphi	**8**
La fiche	9
L'éditeur de code	9
L'inspecteur d'objet	10
Chapitre 3	
Certaines notions de bases	**11**
MessageDlg	11
Routines standards et entrées	13
Certaines propriétés	14
Certaines méthodes	14
Certains évènements	15
Exemple de propriété	15
Exemple d'évènement	17
Chapitre 4	
Comment répondre aux actions de l'utilisateur à l'exécution	**18**
OnKeyPress	18
OnKeyDown	18
OnKeyUp	18
OnMouseMove	20
OnMouseUp	20
OnMouseDown	20
L'évènement TShiftState	21
Chapitre 5	
Certains objets Delhi	**22**
Panel	22
Button	22
MainMenu	23
PopupMenu	23
PageControl	23
ActionList	24
ImageList	26
Image	29
ToolBar	31
Timer	32
Label	32

RadioButton & CheckBox 33
RadioGroup 34
ListBox 35
Edit 36
ComboBox 38
Memo & RichEdit 38
StringGrid 39

Chapitre 6
OnDragOver & OnDragDrop **41**

Chapitre 7
Objets pour manipuler une Base De Données **43**
Accès à une Base De Données (AccesBD) 43
DataSource 43
Table 43
Databse 44
Query 45
Contrôle d'une Base De Données (ControlBD) 45
DBGrid 45
DBNavigator 46
DBText 47
DBEdit 47
DBMemo 48
DBListBox 48
DBComboBox 49
DBLookUpListBox 49
DBLookUpComboBox 50
Quick Report, pour afficher vos tables (QReport) 50
QuickRep 50
QRLabel 51
QRDBText 51
QRExpr 52
QRSysData 52
52

Chapitre 8
Comment effectuer une recherche dans une table ? **53**

Chapitre

1

Introduction

Qu’est-ce que Delphi ?

Delphi est un environnement de programmation visuel orienté objet pour le développement rapide d’applications (RAD). En utilisant Delphi, vous pouvez créer des applications Microsoft Windows 95, Windows 98 et Windows NT très efficaces, avec un minimum de codage manuel. Delphi fournit tous les outils qui vous sont nécessaires pour développer, tester, déboguer et déployer des applications, incluant une importante bibliothèque de composants réutilisables, un ensemble d’outils de conception, des modèles d’applications et de fiches, ainsi que des experts de programmation. Ces outils simplifient le prototypage et réduisent la durée du développement.

Remarque :
Avant de créer une application en Delphi, il faut d’abord créer un nouveau répertoire dans lequel il faut enregistrer le projet ainsi que tous les fichiers associés à ce dernier.

Utilisation du modèle objet

La programmation orientée objet (POO) est une extension de la programmation structurée qui intensifie la réutilisation du code et l’encapsulation de données avec des fonctionnalités. Quand vous avez créé un objet (ou, plus précisément, une classe), vous et d’autres programmeurs pouvez l’utiliser dans d’autres applications ce qui réduit les temps de développement et accroît la productivité.

Qu’est-ce qu’un objet ?

Un objet, ou *class*e, est un type de données qui regroupe des *données* et des *opérations sur ces donnée*s. Avant la programmation orientée objet, les données et les opérations (les fonctions) constituaient des éléments distincts.
Vous pouvez comprendre les objets si vous comprenez les *enregistrements* Pascal Objet. Les enregistrements (analogues aux *structures* en C) sont constitués de champs qui contiennent des données, chaque champ ayant son propre type. Les enregistrements sont un moyen commode de désigner une collection éléments de données variés. Les objets sont également des collections d’éléments de données. Mais les objets, à la différence des enregistrements, contiennent des procédures et fonctions portant sur leurs données. Ces procédures et fonctions sont appelées des *méthode*s.
Les éléments de données d’un objet sont accessibles via des *propriété*s. Les propriétés des objets Delphi ont une valeur qu’il est possible de modifier à la conception sans écrire de code. Si vous voulez modifier la valeur d’une propriété à l’exécution, il vous suffit d’écrire un minimum de code.

La combinaison des données et de fonctionnalités dans un seul élément est appelée *encapsulatio*n. Outre l'encapsulation, la programmation orientée objet est caractérisée par *l'héritage* et le *polymorphisme*. Héritage signifie que les objets dérivent leurs fonctionnalités d'autres objets appelés *ancêtre*s); les objets peuvent modifier leurs comportements hérités. Polymorphisme signifie que différents objets qui dérivent d'un même ancêtre gèrent la même interface de méthode et de propriété, on dit aussi qu'ils sont interchangeables.

Chapitre

2

Environnement Delphi

La barre de taches Delphi

 Voir une unité

 voir une fiche

 Basculer Unité/Fiche

 Nouvelle Fiche

 Exécuter

 Ajouter un fichier au projet

 Retirer un fichier du projet

 Ouvrir un projet

 Tout enregistrer (Le projet et les fichiers associés)

 Enregistrer (Attention : ceci n'enregistre pas le projet)

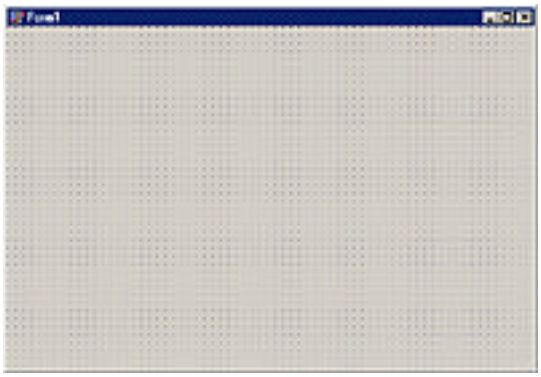
 Ouvrir un fichier

 Nouveau (Liste de choix : Application, Application console, unité, fiche, …etc)

La fiche, L'éditeur de code et l'inspecteur d'objet

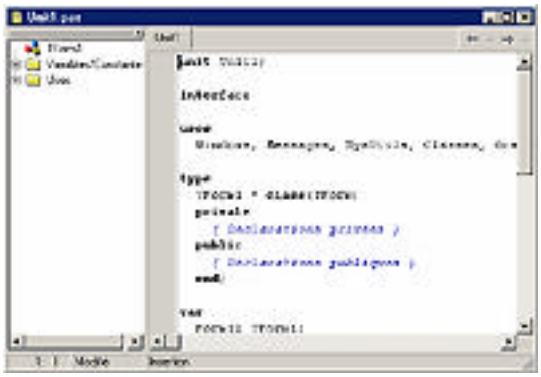
La fiche

Ou *concepteur de fiche* contenant une fiche vierge pour démarrer la conception de l'interface utilisateur de votre application. Une application peut inclure plusieurs fiches.



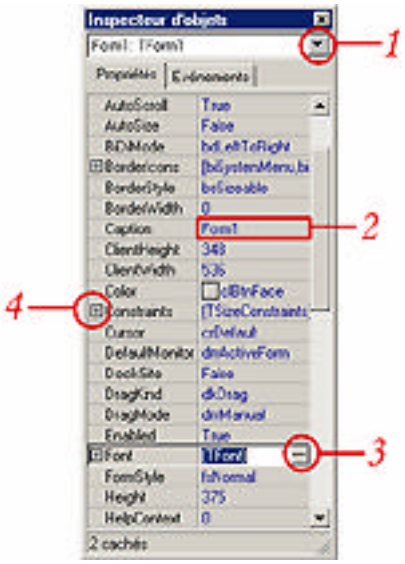
L'éditeur de code

Vous pouvez aussi ajouter directement du code à vos fichiers source en utilisant l'éditeur de code intégré. L'éditeur de code est un éditeur ASCII complet.



L'inspecteur d'objet

Vous pouvez changer la manière dont un composant s'affiche et se comporte dans votre application en utilisant l'inspecteur d'objets. Lorsqu'un composant est sélectionné sur la fiche, ses propriétés et ses événements sont affichés dans l'inspecteur d'objets.



1 - utiliser cette liste déroulante pour sélectionner un objet. Si un objet est sélectionné, ses propriétés seront affichées.
2 - Sélectionnez une propriété et changez sa valeur dans la colonne de droite.
3 - Cliquez sur les points de suspension pour ouvrir une boîte de dialogue permettant de modifier les propriétés d'un objet (si à la place des points de suspension il y'a une flèche vers le bas, alors cliquez dessus pour voir les valeurs valides de la propriété)
4 - double-cliquez sur le signe plus pour ouvrir une liste de détail.

Chapitre

3

Certaines notions de base

Introduction

Pour quitter définitivement l'application (le programme) :

```
Application.Terminate ;
```
Ou:

`Form1.Close;` (si form1 est la fiche principale, sinon, changer form1 par le nom de la fiche principale)

Pour réduire une fiche (fenêtre) : `Application.Minimize ;`

MessageDlg

Affiche une boîte de dialogue de message au centre de l'écran.

```
function MessageDlg(const Msg: string; DlgType: TMsgDlgType;
Buttons: TMsgDlgButtons; HelpCtx: Longint): Word;
```

Description :

Appelez MessageDlg pour afficher une boîte de dialogue de message et gérer la réponse de l'utilisateur. La boîte de dialogue affiche la valeur du paramètre Msg. Utilisez le paramètre DlgType pour indiquer le type de boîte de dialogue. Utilisez le paramètre Boutons pour déterminer quels boutons apparaissent dans le message. Utilisez le paramètre HelpCtx pour spécifier l'ID de contexte de la rubrique d'aide qui doit apparaître quand l'utilisateur clique sur le bouton d'aide ou appuie sur F1 lorsque le dialogue est affiché.
MessageDlg renvoie la valeur du bouton sélectionné par l'utilisateur. Voici les valeurs possibles :

mrAbort	**mrYes**	**mrNone**
mrOk	**mrRetry**	**mrNo**
mrCancel	**mrIgnore**	**mrAll**

Exemple:

Cet exemple utilise un bouton sur une fiche. Quand l'utilisateur clique sur le bouton, une boîte de message demandant à l'utilisateur s'il souhaite quitter l'application apparaît. Si l'utilisateur choisit Oui, une autre boîte de dialogue apparaît pour informer l'utilisateur que l'application est sur le point de se terminer. Quand l'utilisateur choisit OK, l'application se termine.

```
procedure TForm1.Button1Click(Sender: TObject);
begin
  if MessageDlg('Bienvenue dans mon application Pascal Objet.
Quitter?',
    mtConfirmation, [mbYes, mbNo], 0) = mrYes then
  begin
    MessageDlg('Fin de l''application Pascal Objet.', mtInformation,
      [mbOk], 0);
    Close;
  end;
end;
```

Routines standard et Entrées

Beep	Génère un bip standard en utilisant le haut-parleur de l'ordinateur.
Break	Force la sortie d'une instruction **for**, **while** ou **repea**t.
Chr	Renvoie le caractère correspondant à une valeur ASCII spécifiée.
Cos	Calcule le cosinus d'un angle donné.
CurrToStr	Convertit une variable monétaire en chaîne.
Date	Renvoie la date en cours.
DateTimeToStr	Convertit une variable de type *TDateTime* en chaîne.
DateToStr	Convertit une variable de type *TDateTime* en chaîne.
Dec	Décrémente une variable scalaire.
Exit	Sort de la procédure en cours.
Exp	Calcule l'exponentielle d'une valeur donnée.
FloatToStr	Convertit une valeur flottante en chaîne.
FloatToStrF	Convertit une valeur flottante en chaîne en utilisant le format spécifié.
High	Renvoie la plus grande valeur de l'étendue d'un intervalle, d'un tableau ou d'une chaîne.
Inc	Incrémente une variable scalaire.
Int	Renvoie la partie entière d'un nombre réel.
IntToStr	Convertit un entier en chaîne.
Length	Renvoie la longueur d'une chaîne ou d'un tableau.
Low	Renvoie la plus petite valeur de l'étendue d'un intervalle, d'un tableau ou d'une chaîne.
LowerCase	Convertit une chaîne ASCII en minuscules.
MaxIntValue	Renvoie la plus grande valeur signée dans un tableau d'entiers.
MaxValue	Renvoie la plus grande valeur signée dans un tableau.
MinIntValue	Renvoie la plus petite valeur signée dans un tableau d'entiers.
MinValue	Renvoie la plus petite valeur signée dans un tableau.
Now	Renvoie l'heure et la date en cours.
Ord	Renvoie le rang d'une expression de type scalaire.
Pos	Renvoie l'indice dans une chaîne du premier caractère d'une sous-chaîne spécifiée.
Pred	Renvoie le prédécesseur d'une valeur scalaire.
Random	Génère des valeurs aléatoires dans l'intervalle spécifié.
Round	Renvoie la valeur d'un réel arrondie à l'entier le plus proche.
ShowMessage	Affiche une boîte message avec une chaîne non formatée et le bouton OK.
ShowMessageFmt	Affiche une boîte message avec une chaîne formatée et le bouton OK.
Sin	Renvoie le sinus de l'angle spécifié en radians.
Sqr	Renvoie le carré d'un nombre.
Sqrt	Renvoie la racine carrée d'un nombre.
StrToCurr	Convertit une chaîne en valeur monétaire.
StrToDate	Convertit une chaîne en valeur date *(TDateTim*e).
StrToDateTime	Convertit une chaîne en valeur *TDateTim*e.
StrToFloat	Convertit une chaîne en valeur à virgule flottante.
StrToInt	Convertit une chaîne en valeur entière.
StrToTime	Convertit une chaîne au format heure *(TDateTim*e).
StrUpper	Renvoie une chaîne en majuscules.
Succ	Renvoie le successeur d'une valeur scalaire.
Sum	Renvoie la somme des éléments d'un tableau.
Time	Renvoie l'heure en cours.
TimeToStr	Convertit une variable de type *TDateTime* en chaîne.
Trunc	Tronque un nombre réel en un entier.
UpCase	Convertit un caractère en majuscules.
UpperCase	Renvoie une chaîne en majuscules.

Certaines Propriétés

`Alignment`	*Contrôle le positionnement du texte dans le libellé.* *Alignment permet d'indiquer comment le texte du libellé est aligné dans le ClientRect du contrôle libellé.* *L'effet de la propriété Alignment est plus évident si la propriété WordWrap est à True et si le libellé inclut plusieurs lignes de texte.*
`Caption`	de type string, contient la chaîne affichée dans le volet de contrôles (objets)
`Color`	type enum, Contient la couleur de l'objet
`BorderStyle`	type enum, Détermine le style de la bordure d'un objet.
`BorderWidth`	Spécifie la largeur de la bordure du contrôle.
`Font`	Contient les paramètres du texte (soit Caption, ou Text)
`Visible`	type booléen, contient, TRUE : Pour rendre l'objet visible, et FALSE sinon
`Enabled`	type booléen, contient, TRUE : Pour rendre le contrôle de l'objet accessible, et FALSE sinon
`Hint`	Type String, contient la chaîne de texte apparaissant lorsque l'utilisateur déplace la souris au-dessus du contrôle (l'objet).
`ShowHint`	type booléen, contient True pour activer Hint, False sinon.
`TabOrder`	type entier, indique la position du contrôle dans l'ordre de tabulation de son parent. TabOrder n'a de sens que si la propriété TabStop a la valeur True et si le contrôle a un parent (la propriété TabOrder d'une fiche n'a pas de sens sauf si la fiche est l'enfant d'une autre fiche). Un contrôle dont la propriété TabOrder a la valeur -1 se trouve hors de l'ordre de tabulation et ne peut être atteint en utilisant la touche Tab. Pour retirer un contrôle ayant un parent de l'ordre de tabulation initialisez sa propriété TabStop à False.
`PopupMenu`	Identifie le menu surgissant associé au contrôle.
`Heigh`	type entier, détermine la hauteur du contrôle (objet) en pixels
`Left`	type entier, détermine la coordonnée horizontale, exprimée en pixels relativement à la fiche, du bord gauche d'un composant.
`Top`	type entier, détermine la coordonnée verticale, exprimée en pixels relativement à la fiche, du bord gauche d'un composant.
`Width`	type entier, détermine la largeur du contrôle (objet) en pixels
`Cursor`	Spécifie l'image utilisée pour représenter le pointeur de la souris lorsqu'il passe au-dessus de la région couverte par le contrôle.
`Ctl3D`	Détermine si un contrôle a un aspect visuel 2D (deux dimensions) ou 3D (trois dimensions).
`Name`	Contient le nom du composant tel qu'il est désigné dans le code.

Certaines Méthodes

`Close`	pour une fiche principale, Close permet de quitter le programme, sinon, il permet de fermer uniquement la fenêtre
`Show`	
`ShowModal`	Rend un contrôle (objet) visible. ShowModal pour les fenêtres permet de rendre la fenêtre visible, et de désactiver la première
`Hide`	Rend un contrôle invisible

Certains évènements

OnActivate Se produit quand une application devient active.
OnClick Se produit quand l'utilisateur clique sur le contrôle.
OnDblClick Se produit quand l'utilisateur double clique sur le contrôle.
OnCreate Se produit à la création de la fiche.
OnShow Se produit quand la fiche est affichée (c'est-à-dire quand la propriété Visible de la fiche prend la valeur True).
OnHide Se produit quand la fiche est cachée (c'est-à-dire quand la propriété Visible de la fiche prend la valeur False).
OnEnter Se produit quand un contrôle reçoit la focalisation.
OnExit Se produit quand la focalisation passe du contrôle à un autre contrôle.

Exemple de Propriété

La propriété `Align`

```
type TAlignSet = set of TAlign;
type TAlign = (alNone, alTop, alBottom, alLeft, alRight, alClient);
```

TAlignSet est un ensemble de valeurs TAlign. TAlign spécifie comment un contrôle est placé en fonction de son parent. Voici ses valeurs possibles :

Valeur	Signification
`AlNone`	Le contrôle reste à l'emplacement où il a été mis. Valeur par défaut.
`alTop`	Le contrôle se place en haut de son parent et prend toute la largeur de son parent. Sa hauteur n'est pas modifiée.
`alBottom`	Le contrôle se place en bas de son parent et prend toute la largeur de son parent. Sa hauteur n'est pas modifiée.
`alLeft`	Le contrôle se place sur le bord gauche de son parent et prend toute la hauteur de son parent. Sa largeur n'est pas modifiée.
`alRight`	Le contrôle se place sur le bord droit de son parent et prend toute la hauteur de son parent. Sa largeur n'est pas modifiée.
`alClient`	Le contrôle remplit la zone client de son parent. Si un autre contrôle occupe déjà une partie de cette zonne, le contrôle nouveau se redimensionne pour prendre le reste de la zone.

La propriété `Position`

Représente la taille et la position de la fiche.
Utilisez la propriété Position pour connaître ou définir la taille et l'emplacement d'une fiche. Position peut prendre l'une des valeurs suivantes : *poDesigned, poDefault, poDefaultPosOnly, poDefaultSizeOnly, poScreenCenter, poDesktopCenter, poMainFormCenter, poOwnerFormCenter*

Valeur	Signification
`PoDesigned`	La fiche apparaît à l'écran à la position et avec les dimensions qu'elle avait à la conception.
`poDefault`	La fiche apparaît à la position et avec les dimensions déterminées par Windows. A chaque exécution de l'application, la fiche est légèrement déplacée en bas à droite. Le bord droit de la fiche est toujours à proximité du bord de l'écran, et le bord inférieur de la fiche est toujours à proximité du bas de l'écran indépendamment de la résolution de l'écran.
`poDefaultPosOnly`	La fiche apparaît avec la taille utilisée à la conception, mais Windows choisit sa position à l'écran. A chaque exécution de l'application, la fiche est légèrement déplacée en bas à droite. Quand la fiche ne peut plus se déplacer vers le bas et la droite tout en conservant sa taille et en étant affichée entièrement, la fiche apparaît dans l'angle supérieur gauche de l'écran.
`poDefaultSizeOnly`	La fiche apparaît à la position utilisée à la conception, mais Windows choisit sa taille. Le bord droit de la fiche est toujours à proximité du bord de l'écran, et le bord inférieur de la fiche est toujours à proximité du bas de l'écran indépendamment de la résolution de l'écran.
`poScreenCenter`	La fiche conserve les dimensions définies à la conception, mais est positionnée au centre de l'écran. Dans les applications multi-écran, la fiche peut être déplacée de cette position centrale afin de tenir complètement sur un écran, comme spécifié par la propriété DefaultMonitor.
`poDesktopCenter`	La fiche conserve la taille définie à la conception mais elle est placée au centre de l'écran. Il n'y a pas d'ajustements pour les applications multi-écran.
`poMainFormCenter`	La fiche conserve la taille que vous lui aviez donné à la conception, mais est placée dans le centre de la fiche principale de l'application. Aucun ajustement n'est réalisé pour applications multi-moniteurs. Cette position ne doit être utilisé qu'avec des fiches secondaires. Si elle est définie pour une fiche principale, elle agit comme poScreenCenter.
`poOwnerFormCenter`	La fiche conserve la taille que vous lui aviez donné à la conception, mais est placée dans le centre de la fiche spécifiée par la propriété Owner. Si la propriété Owner ne spécifie pas de fiche, cette position agit comme poMainFormCenter.

Exemple d'événement

L'événement `OnClose`

Se produit quand la fiche se ferme.

```
type
  TCloseAction = (caNone, caHide, caFree, caMinimize);
  TCloseEvent = procedure(Sender: TObject; var Action: TCloseAction)
of object;
```

Utilisez l'événement OnClose pour effectuer des actions particulières quand la fiche se ferme. L'événement OnClose spécifie le gestionnaire d'événement à appeler quand une fiche va être fermée. Le gestionnaire spécifié par OnClose peut, par exemple, s'assurer que tous les champs d'une fiche de saisie ont un contenu correct avant d'autoriser la fermeture de la fiche.
Une fiche est fermée par la méthode Close ou si l'utilisateur choisit la commande Fermeture dans le menu système de la fiche.
Le type TCloseEvent pointe sur une méthode qui gère la fermeture d'une fiche. La valeur du paramètre ***Action*** détermine si la fiche se ferme effectivement. Les valeurs possibles de Action sont :

Valeur	Signification
caNone	La fiche n'est pas autorisée à se fermer, il ne se passe rien.
CaHide	La fiche n'est pas fermée, juste cachée. L'application peut toujours accéder à la fiche.
CaFree	La fiche est fermée et toute la mémoire allouée à la fiche est libérée.
caMinimize	La fiche est réduite, pas fermée. C'est l'action par défaut pour les fiches enfant MDI.

Si une fiche est un enfant MDI alors que sa propriété BorderIcons a la valeur biMinimize, l'action par défaut est caMinimize. Si une fiche enfant MDI n'a pas ce paramètre, l'action par défaut est caNone, c'est-à-dire que rien ne se produit quand l'utilisateur tente de fermer la fiche.
Si une fiche est une fiche enfant SDI, l'action par défaut est caHide.
Pour fermer une fiche et la libérer dans un événement OnClose, affectez la valeur caFree à Action.

Exemple :
Cet exemple affiche une boîte message quand l'utilisateur tente de fermer la fiche. Si l'utilisateur clique sur le bouton Oui, la fiche se ferme sinon elle reste ouverte.

```
procedure TForm1.FormClose(Sender: TObject; var Action:
TCloseAction);
begin
  if MessageDlg('Fermer application ?', mtConfirmation,
    [mbYes, mbNo], 0) = mrYes then
    Action := caFree
  else
    Action := caNone;
end;
```

Chapitre

4

Comment répondre aux actions de l'utilisateur à l'exécution

Les actions lancées lorsqu'un utilisateur appuie sur une touche ou une combinaison de touches du clavier sont :

OnKeyPress

Se produit quand un utilisateur appuie sur une touche alphanumérique. Les touches n'ayant pas d'équivalent ASCII (Maj ou F1, par exemple) ne génèrent pas d'événement OnKeyPress

Exemple :
Ce gestionnaire d'événement affiche une boîte de dialogue spécifiant la touche qui a été appuyée :

```
procedure TForm1.FormKeyPress(Sender: TObject; var Key: Char);
begin
  if Key = 'k' then ShowMessage('oui')
end;
```

OnKeyDown

Se produit quand l'utilisateur appuie sur une touche alors que le contrôle détient la focalisation.

OnKeyUp

Se produit quand l'utilisateur relâche une touche enfoncée
Exemple :
Le code suivant arrête un travail d'impression lorsque l'utilisateur appuie sur Echap. Remarquez que vous pourriez initialiser KeyPreview à True pour vous assurer que le gestionnaire d'événement OnKeyDown de Form1 est appelé.

```
procedure TForm1.FormKeyDown(Sender: TObject; var Key: Word; Shift:
TShiftState);
begin
if (Key = VK_ESCAPE) then then ShowMessage('oui')
end;
```

TShiftState indique l'état des touches Alt, Ctrl, Maj et des boutons de la souris.
Le type TShiftState est utilisé par les gestionnaires du clavier et de la souris pour déterminer l'état des touches Alt, Ctrl et Maj ainsi que l'état des boutons de la souris au moment où l'événement se produit. Consulter le chapitre suivant pour voir l'ensemble des indicateurs et leur signification.

Key contient :

VK_LBUTTON	bouton Gauche Souris	**VK_CANCEL**	Arrêt Exec. Programme
VK_RBUTTON	bouton Droite Souris	**VK_BACK**	Retour Arrière
VK_MBUTTON	bouton Central Souris	**VK_TAB**	Tabulation
VK_RETURN	Entrée	**VK_SHIFT**	Touche de contrôle MAJ
VK_CONTROL	Touche de contrôle CTRL	**VK_MENU**	Touche de contrôle ALT
VK_PAUSE	Pause	**VK_ESCAPE**	Echappemsnt
VK_SPACE	Barre d'espacement	**VK_PRIOR**	Page Haut
VK_NEXT	Page Bas	**VK_END**	Fin
VK_HOME	Début	**VK_LEFT**	Flèche gauche
VK_UP	Flèche haut	**VK_RIGHT**	Flèche droite
VK_DOWN	Flèche bas	**VK_SNAPSHOT**	Impression d'écran
VK_INSERT	Insérer	**VK_DELETE**	Supprimer
VK_HELP	Aide	**VK_NUMPAD0..VK_NUMPAD9**	Touche pavé numérique 0 à 9
VK_MULTIPLY	Touche pavé numérique *****	**VK_ADD**	Touche pavé numérique **+**
VK_SEPARATOR	Touche [Entrée]	**VK_SUBTRACT**	Touche pavé numérique **-**
VK_DECIMAL	. (Point décimal)	**VK_DIVIDE**	Touche pavé numérique **/**
VK_F1..VK_F12	Touches de fonctionF1 à F12	**VK_NUMLOCK**	Verrouillage pavé numérique
VK_SCROLL	Verrouillage scrolling	**VK_CAPITAL**	Verrouillage majuscules

Tableau1

Remarque :
Le Key dans OnKeyPress et celui dans OnKeyDown/OnKeyUp n'est pas de même type. Le premier est de type char (caractère) et l'autre de type Word (voir tableau 1)

Les actions lancées lorsqu'un utilisateur clique sur un bouton de la souris sont :

OnMouseMove

Se produit quand l'utilisateur déplace le pointeur de la souris au-dessus d'un contrôle.

```
type TMouseMoveEvent = procedure(Sender: TObject; Shift:
TShiftState; X, Y: Integer) of object;
```

Utilisez le gestionnaire d'événement OnMouseMove pour répondre lorsque le pointeur de la souris se déplace après que le contrôle ait capturé la souris.
Utilisez le paramètre Shift du gestionnaire d'événement OnMouseMove pour déterminer l'état des touches mortes et des boutons de la souris. Les touches mortes sont les touches Maj, Ctrl et Alt ou des combinaisons de touches mortes et des boutons de la souris. X et Y indiquent les coordonnées, exprimées en pixels, du pointeur de la souris dans la zone client de Sender.

Exemple :

```
procedure TForm1.FormMouseMove(Sender: TObject; Shift: TShiftState;
X, Y: Integer);
begin
 Label1.Caption : = IntToStr(x) ;
 Label2.Caption : = IntToStr(y) ;
end ;
```

OnMouseUp

Se produit lorsque l'utilisateur relâche un bouton de la souris qui a été enfoncé alors que le pointeur de la souris se trouvait au-dessus d'un composant.

OnMouseDown

Se produit quand un utilisateur appuie sur un bouton de la souris alors que le pointeur de la souris est au-dessus d'un contrôle.

```
type TMouseEvent = procedure (Sender: TObject; Button: TMouseButton;
                    Shift: TShiftState; X, Y: Integer) of object;
```

TMouseButton définit les constantes de bouton de la souris utilisées par les gestionnaires d'événements.

```
type TMouseButton = ( mbLeft , mbRight , mbMiddle );
```

Le type TMouseButton définit les constantes de bouton de la souris utilisées par les gestionnaires d'événements de la souris pour connaître le bouton à l'origine de l'événement.
Le type TShiftState est utilisé par les gestionnaires du clavier et de la souris pour déterminer l'état des touches Alt, Ctrl et Maj ainsi que l'état des boutons de la souris au moment où l'événement se produit. Voici un ensemble d'indicateurs et leur signification :

Le type **TshiftState**

Valeur	*Signification*
ssShift	La touche Maj est enfoncée.
ssAlt	La touche Alt est enfoncée.
ssCtrl	La touche Ctrl est enfoncée.
SsLeft	Le bouton gauche de la souris est enfoncé.
ssRight	Le bouton droit de la souris est enfoncé.
ssMiddle	Le bouton central de la souris est enfoncé.
ssDouble	On a double-cliqué sur la souris.

Exemple :

```
procedure TForm1.FormMouseDown(Sender: TObject; Button:
TMouseButton;
  Shift: TShiftState; X, Y: Integer);
begin
 if button=mbright then showmessage('oui')
end;
```

Chapitre

5

Certains objets Delphi

 Panel

Utilisez TPanel pour placer un volet vide dans une fiche. Les volets disposent de propriétés permettant d'entourer le contrôle d'une bordure biseautée, ainsi que des méthodes facilitant la gestion du positionnement des contrôles enfant incorporés dans le volet.

Proprété :

`BevelInner`	Détermine le style du biseau interne d'un volet.
`BevelOuter`	Détermine le style du biseau externe d'un volet.
`BevelWidth`	Spécifie la distance, exprimée en pixels, qui sépare les biseaux interne et externe.

Voir `BorderStyle, BorderWidth` dans le chapitre *Certaines Propriétés*

 Button

Utilisez TButton pour placer un bouton poussoir Windows standard dans une fiche. TButton introduit plusieurs propriétés permettant de contrôler son comportement dans la définition d'une boite de dialogue. Les utilisateurs choisissent des contrôles bouton pour effectuer une action.

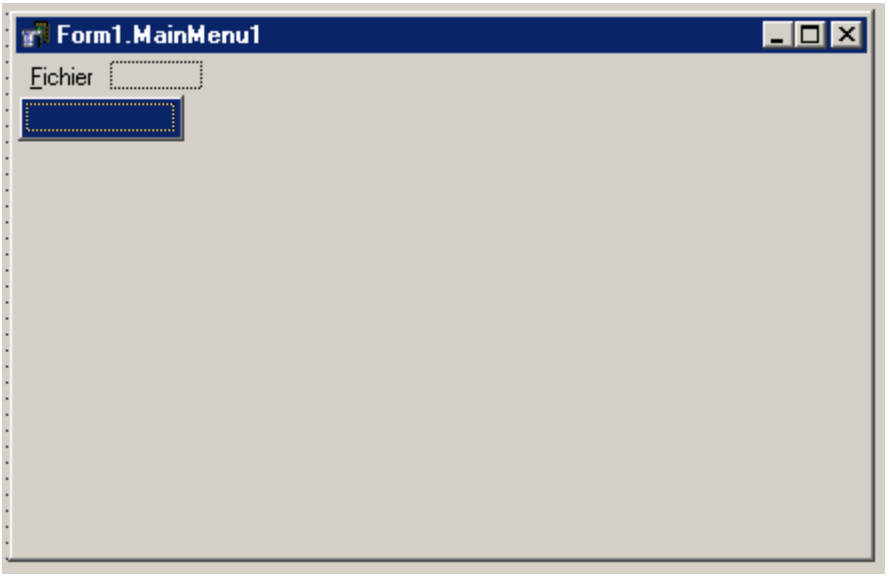
Pour utiliser un bouton affichant un bitmap au lieu d'un texte, utiliser TBitBtn. Pour utiliser un bouton pouvant rester en position enfoncée, utilisez TSpeedButton.

Voir :
La propriété `Cursor`, et l'évènement `OnClick` dans le chapitre *Certaines Propriétés* et *Certains évènements.*

Remarque :
L'évènement OnClick pour un bouton est très utilisé, pour accéder directement à la procédure sans passer par l'inspecteur d'objet, double cliquer sur l'objet bouton qui est sur la fiche.

MainMenu

Utilisez TMainMenu pour placer un menu principal dans une fiche. Pour commencer la conception d'un menu, ajoutez un composant menu principal à votre fiche, puis double-cliquez sur le composant.



PopupMenu

Utilisez TPopupMenu pour définir le menu surgissant qui apparaît quand l'utilisateur clique sur un contrôle avec le bouton droit de la souris. Pour qu'un menu surgissant soit disponible, affectez l'objet TPopupMenu à la propriété PopupMenu du contrôle.
La conceptions d'un PopupMenu se fait éxactement comme celle d'un MainMenu.

PageControl

Utilisez TPageControl pour créer une boîte de dialogue multipage ou un classeur à onglets. TPageControl affiche plusieurs pages superposées qui sont des objets TTabSheet. L'utilisateur sélectionne une page en cliquant sur l'onglet de la page qui apparaît en haut du contrôle. A la conception, pour ajouter une nouvelle page à un objet TPageControl, cliquez avec le bouton droit de la souris dans l'objet TPageControl et choisissez Nouvelle page.

`ActivePage` Spécifie la page affichée dans le contrôle pages.

MultiLine	Détermine si les onglets peuvent apparaître sur plusieurs lignes.
Style	Spécifie le style du contrôle onglets.

Valeur	Signification
TsTabs	Onglets standard
tsButtons	Onglets boutons
tsFlatButtons	Onglets boutons plats

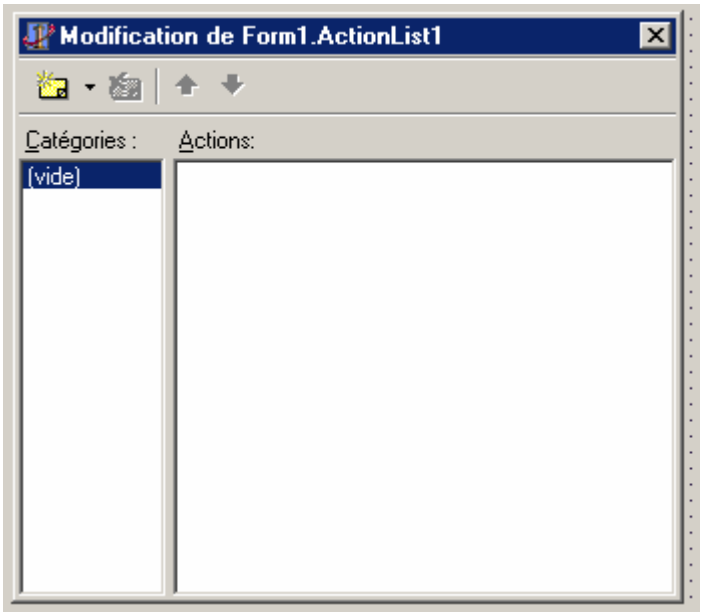
TabPosition	Détermine si les onglets apparaissent en haut ,en bas, à gouche, ou à droite.
TabWidth	Indique la largeur, exprimée en pixels, des onglets d'un contrôle onglets.

TabSheet :
c'est une page individuelle d'un objet TPageControl.
Pour donner un titre pour une page (TabSheet) dans son onglet, utiliser la propriété Caption

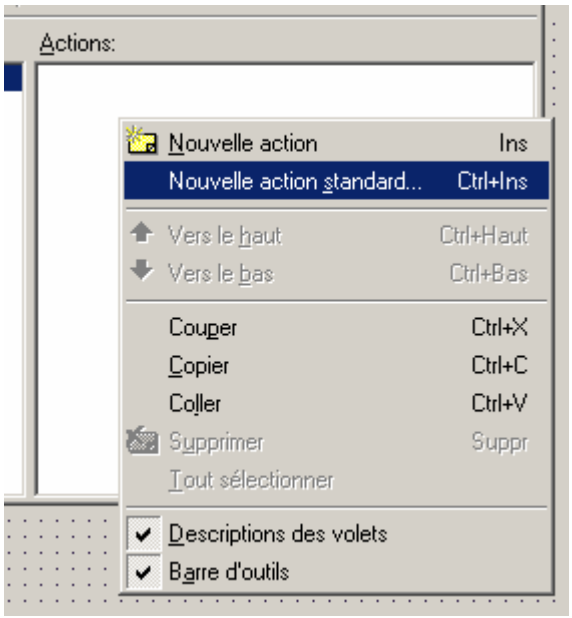
ActionList

TActionList implémente le comportement générique introduit dans CustomActionList, mais n'introduit aucun comportement nouveau. TactionList publie quelques membres hérités de TCustomActionList.
Pour centraliser la réponse à des commandes utilisateur (actions), utilisez des listes d'actions, conjointement avec des actions et des liaisons d'actions. Les composants des listes d'actions sont l'interface utilisateur permettant de travailler avec des actions. Ajoutez des composants de listes d'actions à votre fiche ou à votre module de données à partir de la page standard de la palette des composants. Pour afficher l'éditeur de liste d'actions à partir duquel vous pouvez ajouter, supprimer et réorganiser les actions, double-cliquez sur la liste d'actions.

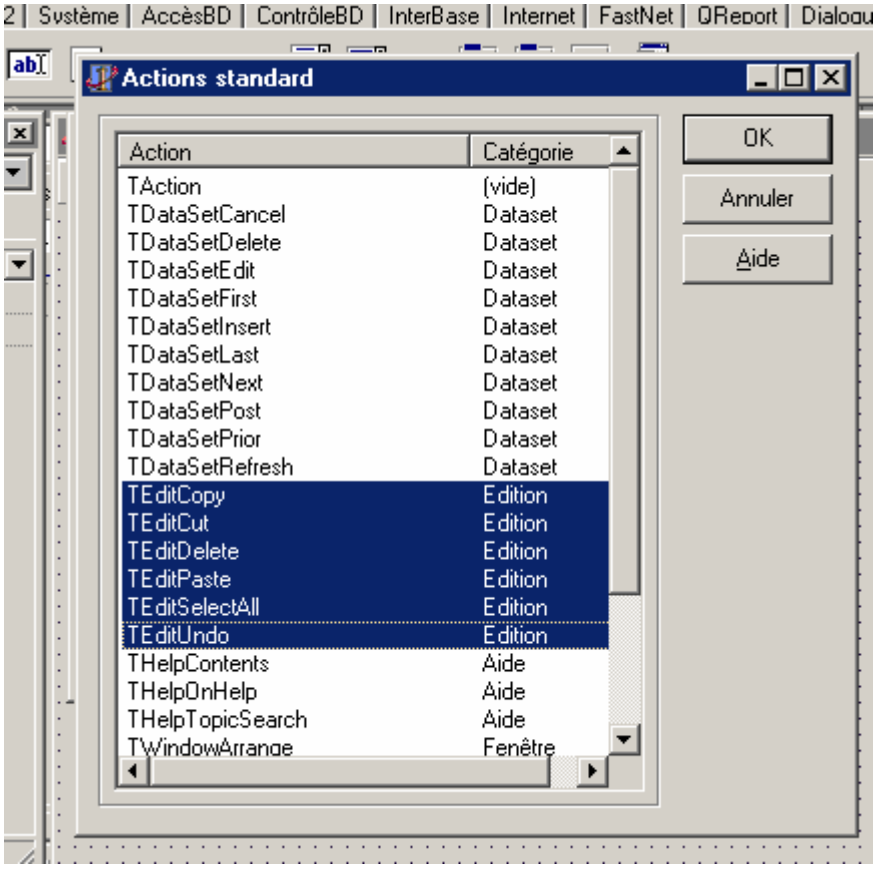
Pour créer des actions standards :
1 - Aouter un composant Imagelist
2 - Modifier la propriété Images du composant ActionList : donner le nom du composant ImageList ajouté, 'ImageList1'.
3 - Double cliquez sur l'objet ActionList :



4 - Dans la partie Actions (à droite), cliquez sur le bouton droit de la souris :



4 - Cliquez maintenant sur Nouvelle action standard…



5 - Choisissez les actions voulues, puis cliquez sur OK.

ImageList

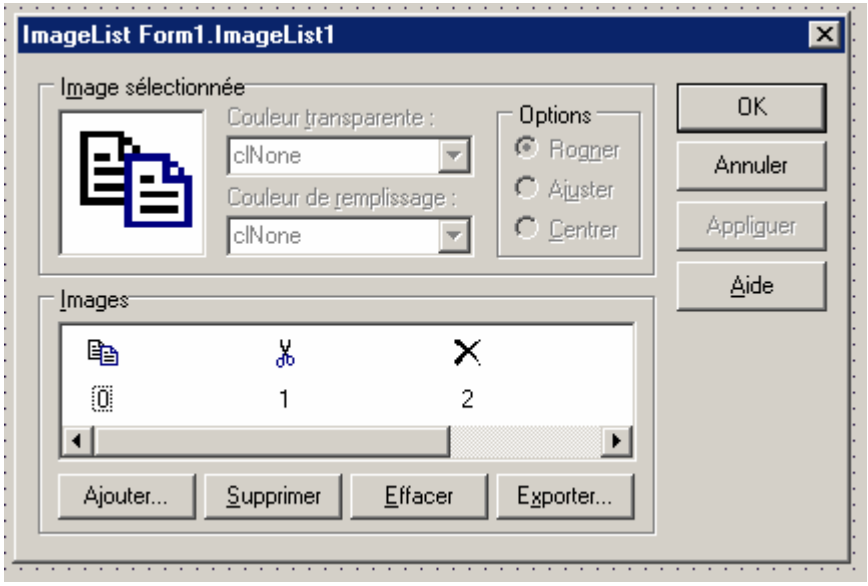
Remarque :
Si vous utilisez un objet ActionList, les images de ImageList son ajoutées automatiquement, sinon vous pouvez ajouter vous même des images de votre choix.

Les listes d'images sont utilisées pour gérer de manière efficace de grands nombres d'icônes ou de bitmaps. Toutes les images d'une liste d'images sont contenues dans un seul grand bitmap au format de périphérique écran. Une liste d'images peut également contenir un bitmap monochrome contenant des masques à utiliser pour dessiner des images en transparence (le style icône).

Une liste d'images peut contenir de nombreuses images de même taille et permet d'accéder aux images via un indice dans l'intervalle 0 à n - 1. La liste d'images dispose de méthodes pour faciliter le stockage, la lecture et le dessin des images stockées.

Pour ajouter des images à une liste d'images lors de la conception, cliquez sur le bouton droit et choisissez Editeur de liste d'images.

L’editeur ImageList :



En mode conception, vous utilisez l'éditeur de liste d'images pour ajouter des bitmaps et des icônes à un composant TImageList.
Lorsque vous travaillez dans l'éditeur de liste d'images, vous pouvez cliquer sur Appliquer pour enregistrer votre travail sans quitter l'éditeur, ou cliquer sur OK pour enregistrer vos modifications et quitter le dialogue. L'utilisation du bouton Appliquer est particulièrement intéressante car, une fois que vous avez quitté le dialogue, vous ne pouvez plus modifier les images existantes.

Pour afficher l'éditeur de liste d'images :
Sélectionnez l'objet TImageList et double-cliquez sur le composant ou cliquez avec le bouton droit et sélectionnez Editeur ImageList.

Image sélectionnée
Ce contrôle affiche l'image sélectionnée. Vous pouvez changer cette image en cliquant sur une autre image de la liste Images qui est au-dessous. Quand une image est sélectionnée, vous pouvez la supprimer de la liste des images. Si l'image n'a pas été ajoutée à la liste des images avant la session en cours de l'éditeur, vous pouvez utiliser les autres contrôles pour modifier ses propriétés. Cependant, une fois que l'éditeur de liste d'images est fermé, ces propriétés sont immuables et les contrôles de l'image sélectionnée sont estompés si l'éditeur de liste d'images est rouvert et cette image sélectionnée à nouveau.

Couleur transparente
Utilisez la liste déroulante Couleur transparente pour spécifier la couleur utilisée pour créer le masque servant à dessiner l'image avec transparence. La couleur transparente par défaut est la couleur du pixel du coin inférieur gauche sur le bitmap. Vous pouvez aussi changer la couleur transparente en cliquant directement sur un pixel dans l'image sélectionnée.

Quand une image a une couleur transparente, tous les pixels de l'image ayant cette couleur ne s'affichent pas dans cette couleur, mais sont au contraire transparents et laissent voir ce qui est derrière l'image.

Pour les images icône, la couleur transparente est mise à clNone (les icônes sont déjà masquées).

Couleur remplissage
Utilisez la liste déroulante Couleur remplissage pour spécifier la couleur utilisée ajoutée autour de l'image sélectionnée si elle est plus petite que les dimensions indiquées par les propriétés Height et Width du contrôle liste d'images.

Ce contrôle est estompé si l'image sélectionnée remplit entièrement les dimensions spécifiées par la liste d'images (c'est-à-dire si elle est au moins aussi haute et large que les valeurs des propriétés Height et Width). Ce contrôle est également estompé pour les images icône, car ces dernières jouent le rôle de masques et leur contour est transparent.

Options
Utilisez les boutons radio Options pour indiquer comment la liste d'images affichera l'image sélectionnée si elle ne correspond pas exactement aux dimensions spécifiées par les propriétés Height et Width de la liste d'images. Ces boutons sont désactivés pour les icônes.)

Option	Description
Rogner	Affiche une partie de l'image en partant du coin supérieur gauche, et en étendant la largeur et la hauteur de la liste d'images vers le coin inférieur droit.
Ajuster	Provoque l'étirement de toute l'image afin qu'elle tienne en largeur et en hauteur dans la liste d'images.
Centrer	Centre l'image sur la largeur et la hauteur de la liste d'images. Si la largeur ou la hauteur de l'image est supérieure à la largeur ou à la hauteur de la liste d'images, l'image risque d'être rognée.

Images
Affiche une prévisualisation des images contenues dans la liste d'images. Contient des contrôles permettant d'ajouter ou de supprimer des images dans la liste. Chaque image est affichée dans une zone 24x24 ; il est ainsi possible de visualiser plusieurs images à la fois. Sous chaque image, une légende indique la position de l'image dans la liste (en partant du numéro zéro). Il est possible de modifier la légende de l'image afin de changer sa position dans la liste. Vous pouvez aussi faire glisser l'image à sa nouvelle position.

Ajouter
Affiche la boîte de dialogue Ajouter des images. Elle vous permet de sélectionner un ou plusieurs bitmaps et icônes à ajouter dans la liste d'images. Les images sont ensuite affichées en surbrillance dans la liste et leur légende est numérotée en fonction de leur position dans la liste.
Si un bitmap dépasse la largeur ou la hauteur de la liste d'image, un message vous demande si vous souhaitez que le bitmap soit divisé en plusieurs images. Ceci est particulièrement utile pour les bitmaps de la barre d'outils qui sont souvent composés de plusieurs petites images affichées dans un ordre précis et stockées sous forme d'un bitmap plus grand.

Supprimer
Supprime de la liste d'images les images sélectionnées. Les images restantes sont repositionnées et renumérotées en partant de zéro.
Affiche la boîte de dialogue Ajouter des images. Elle vous permet de sélectionner un ou plusieurs bitmaps et icônes à ajouter dans la liste d'images. Les images sont

ensuite affichées en surbrillance dans la liste et leur légende est numérotée en fonction de leur position dans la liste.
Si un bitmap dépasse la largeur ou la hauteur de la liste d'image, un message vous demande si vous souhaitez que le bitmap soit divisé en plusieurs images. Ceci est particulièrement utile pour les bitmaps de la barre d'outils qui sont souvent composés de plusieurs petites images affichées dans un ordre précis et stockées sous forme d'un bitmap plus grand.

Supprimer
Supprime de la liste d'images les images sélectionnées. Les images restantes sont repositionnées et renumérotées en partant de zéro.

Exporter
Vous permet d'enregistrer dans un fichier l'image sélectionnée. Ce fichier contient le bitmap tel quel, avec les modifications (rognage ou étirement) effectuées.

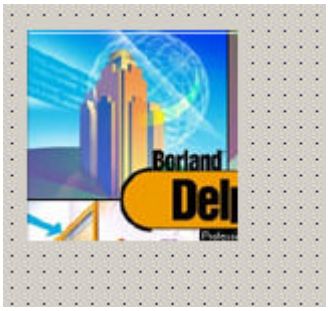
Image

Utilisez TImage pour afficher une image graphique dans une fiche. Utilisez un objet TPicture pour spécifier l'objet bitmap, icône, métafichier ou autre qui est affiché par TImage. TImage introduit diverses propriétés permettant de déterminer comment l'image est affichée à l'intérieur de l'objet TImage.

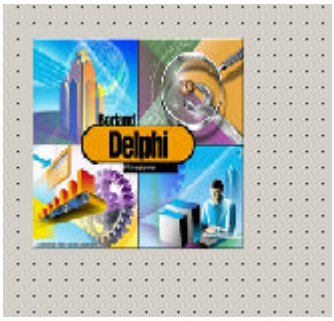
Pour afficher une image dans un composant Image :
1 - Double cliquez sur le composant Image qui se trouve dans la fiche :

2 - Cliquez sur le bouton Charger…

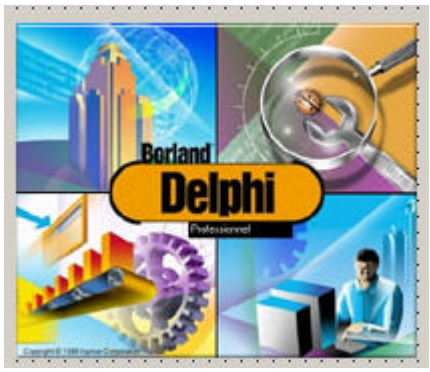
3 - Choisissez une image qui se trouve dans votre disque dûr.



4 - Si l'image est grande, et qu'elle n'apparaît pas complètement dans votre composant Image:
Modifier la propriété ***Stretch*** à True, si vous volez donner des dimensions de votre choix.



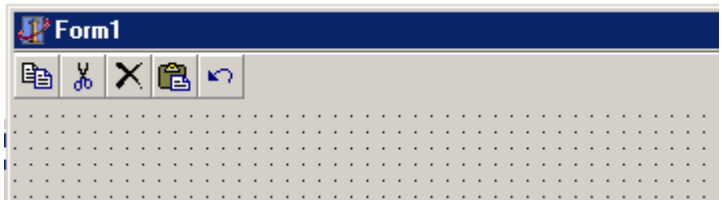
5 - sinon, modifier la propriété ***AutoSize*** à true, si vous voulez garder les dimensions réelles de l'image.



Les proprétés :

`AutoSize`	Spécifie si le contrôle se redimensionne automatiquement pour s'adapter à son contenu.
`Center`	Indique si l'image est centrée dans le contrôle image.
`Stretch`	Indique si l'image doit être modifiée afin qu'elle rentre exactement dans les limites du contrôle image.

ToolBar



TToolBar est un conteneur pour les boutons d'outils (TToolButton). Il offre un moyen simple d'organiser et de gérer des contrôles visuels.

Tous les boutons d'outils d'une barre d'outils conservent une largeur et une hauteur uniformes.

Une barre d'outils peut contenir d'autres contrôles. Ces contrôles (maintenus en place par des boutons d'outils invisibles) conservent une hauteur uniforme.

Les contrôles peuvent passer automatiquement à la ligne suivante s'ils ne tiennent pas horizontalement sur la barre d'outils.

La propriété Flat permet de faire apparaître l'arrière-plan derrière la barre d'outils et d'affecter des bordures apparentes aux boutons d'outils.

Des espaces et des séparateurs (qui sont en fait des boutons d'outils configurés spécialement) peuvent grouper les contrôles sur la barre d'outils à la fois visuellement et fonctionnellement.

Habituellement, les boutons d'outils correspondent aux éléments de menus de l'application et ils permettent à l'utilisateur d'accéder d'une manière plus directe aux commandes de l'application.

Les propriétés :

Flat	Rend la barre d'outils transparente et élimine les bordures pour les boutons d'outils.
Images	Enumère les images pouvant apparaître sur des boutons d'outils.

Timer

TTimer est utilisé pour simplifier l'appel des fonctions SetTimer et KillTimer de l'API Windows et le traitement des messages WM_TIMER. Utilisez un composant timer pour chaque timer de l'application.

Les propriétés et événements du composant timer affectent le timer en spécifiant des informations sur l'événement timer (parmi ces informations, l'intervalle du timer qui correspond au paramètre de la fonction SetTimer de l'API Windows). L'exécution du timer se produit réellement au travers de son événement OnTimer.

Les propriétés :

Enabled	Détermine si le timer répond aux événements timer.
Interval	Détermine l'intervalle de temps, exprimé en millisecondes, s'écoulant avant que le composant timer génère un autre événement OnTimer.

Label

TLabel est un contrôle non-fenêtré qui affiche du texte dans une fiche. Ce texte peut être utilisé pour libeller un autre contrôle et peut attribuer la focalisation à ce contrôle quand l'utilisateur saisit un raccourci clavier.

Propriétés :

AutoSize

type booléen:

AutoSize permet de faire en sorte que le libellé ajuste sa taille automatiquement afin que la zone client s'adapte à la hauteur et à la largeur du texte. Quand AutoSize est à False, le libellé a une taille fixe. Quand AutoSize est à True, la taille du libellé est réajustée quand son texte change. La taille du libellé est aussi réajustée quand la propriété Font change.

Quand WordWrap est à True, la largeur du libellé est fixe. Si AutoSize est aussi à True, les changements apportés au texte provoquent un changement de hauteur du

libellé. Quand AutoSize est à True alors que WordWrap est à False, la hauteur du libellé est déterminée par la fonte et les changements apportés au texte provoquent un changement de largeur du libellé.

FocusControl
Affectez à FocusControl le contrôle fenêtré destiné à recevoir la focalisation quand l'utilisateur appuie sur la touche raccourci spécifiée par le libellé. Pour spécifier une touche raccourci, faites précéder un caractère du texte du libellé d'un "et" commercial (&) et initialisez la propriété *ShowAccelChar* à True.

Transparent
type booléen:
Initialisez Transparent à True pour que le libellé n'empêche pas de voir d'autres contrôles sur la fiche. Si, par exemple, le libellé est utilisé pour ajouter du texte à un graphique, vous pouvez initialiser Transparent à True pour que le libellé ne soit pas séparé de l'objet graphique.

L'écriture d'un texte avec un arrière-plan transparent est plus lente que lorsque Transparent est à False. Si l'image placée sous le libellé n'est pas trop complexe, vous pouvez améliorer les performances en choisissant comme couleur d'arrière-plan pour le libellé une couleur correspondant à l'objet situé en dessous et en initialisant Transparent à False.

WordWrap
type booléen:
Initialisez WordWrap à True pour permettre au libellé d'afficher plusieurs lignes de texte. Quand WordWrap est à True, tout texte trop long pour la largeur du contrôle libellé passe à la ligne suivante au niveau de la marge droite et continue sur les lignes suivantes.

Initialisez WordWrap à False pour limiter le libellé à une seule ligne. Quand WordWrap est à False, tout texte trop long pour le libellé est tronqué.

 RadioButton &  CheckBox

RadioButton
Utilisez TRadioButton pour ajouter un bouton radio à une fiche. Les boutons radio proposent à l'utilisateur un ensemble d'options mutuellement exclusives, c'est-à-dire qu'un seul bouton radio d'un groupe peut être sélectionné à la fois à un moment donné. Quand l'utilisateur sélectionne un bouton radio, le bouton radio précédemment sélectionné devient désélectionné. Les boutons radio sont fréquemment regroupés dans une boîte groupe (TGroupBox). Commencez par ajouter la boîte groupe à la fiche, puis choisissez les boutons radio dans la palette des composants et placez-les dans la boîte groupe.

Par défaut, tous les boutons radio placés dans le même contrôle fenêtré conteneur (comme TRadioGroup ou TPanel) appartiennent au même groupe. Par exemple, deux boutons radio d'une fiche ne peuvent être sélectionnés simultanément que s'ils appartiennent à des conteneurs différents, par exemple deux boîtes groupe.

CheckBox
Un composant TCheckBox propose une option à l'utilisateur. L'utilisateur peut activer la case à cocher pour sélectionner l'option, ou supprimer la coche pour la désélectionner.

Ptopriété ;
`Checked`

type booléen:
Consultez la propriété Checked pour déterminer si le bouton radio est sélectionné. Affectez la valeur True à la propriété Checked pour sélectionner le bouton radio et désélectionner tous les autres boutons radio du même conteneur. Affectez la valeur False à Checked pour désélectionner le bouton radio, plus aucun bouton du groupe n'étant alors sélectionné.

RadioGroup

Un objet TRadioGroup est une boîte groupe particulière ne pouvant contenir que des boutons radio. Les boutons radio contenus directement dans le même composant sont "groupés". Quand l'utilisateur active un bouton radio, tous les autres boutons radio de son groupe deviennent désactivés. Il ne peut donc y avoir deux boutons radio activés simultanément dans une fiche que s'ils sont placés dans des conteneurs distincts, comme des boîtes groupe.

Pour ajouter des boutons radio à un objet TRadioGroup, modifiez la propriété Items dans l'inspecteur d'objets. Chaque chaîne de Items fait apparaître un bouton radio dans le groupe ayant la chaîne comme libellé. La valeur de la propriété ItemIndex détermine le bouton radio sélectionné du groupe.

Vous pouvez afficher les boutons radio sur une ou plusieurs colonnes en définissant la valeur de la propriété Columns.

Propriétés :
`Columns`

type entier:
La propriété Columns détermine le nombre de colonnes du groupe de boutons radio. Sa valeur peut varier de 1 à 16. La valeur par défaut est 1, ce qui signifie que les boutons radio sont disposés verticalement sur une seule colonne.

`Items`

type TString:
La propriété Items contient un objet TStrings qui liste les libellés des boutons radio du groupe. Dans les descendants de TCustomRadioGroup, TRadioGroup et TDBRadioGroup, dans lesquels cette propriété est publiée, il est possible d'ajouter ou de supprimer des boutons en modifiant la propriété liste Items avec l'inspecteur d'objets.

Remarque : Les éléments d'un objet groupe de boutons radio sont des instances particulières de TRadioButton générées par TCustomRadioGroup. Il n'est pas

possible d'inclure dans le groupe de boutons radio des instances de TRadioButton créées de manière autonome.

`ItemIndex`

type entier:
La propriété ItemIndex contient l'indice du bouton radio dans la propriété Items. Le premier bouton a l'indice 0. A l'exécution, la valeur de la propriété ItemIndex change quand l'utilisateur sélectionne un bouton radio. Pour qu'un bouton apparaisse sélectionné au démarrage de l'application, affectez à la conception l'indice de ce bouton à la propriété ItemIndex. Sinon, laissez la valeur par défaut de ItemIndex, -1, ce qui signifie qu'il n'y a pas de bouton sélectionné.

ListBox

Utilisez TListBox pour afficher une liste défilante d'éléments où l'utilisateur peut sélectionner, ajouter ou supprimer des éléments.

Propriétés :

`Columns`

type entier:
Utilisez la propriété Columns pour spécifier le nombre de colonnes, dans une boîte liste multicolonne, visibles sans utiliser la barre de défilement horizontale.

`Items`

type TString:
Utilisez la propriété Items pour ajouter, insérer, supprimer ou déplacer des éléments. Par défaut, les éléments d'une boîte liste sont de type TStrings. Utilisez ce type d'élément pour accéder aux propriétés et méthodes permettant de manipuler les éléments de la liste.

Les méthodes Add et Delete :

`Items.add('chaine de caractères')`	ajoute une chaîne en fin de liste.
`Items.Delete(i :integer)`	supprime la chaîne qui se trouve à la $(i+1)^{ème}$ ligne.

Exemple :
le code suivant ajoute un texte comme élément d'une boîte liste :

```
procedure TForm1.FormCreate(Sender: TObject);
begin
  ListBox1.Items.Add('Vertical');
  ListBox1.Items.Add('Horizontal');
end;
```

`MultiSelect`

type booléen:
Affectez la valeur True à la propriété MultiSelect pour permettre à l'utilisateur de sélectionner plusieurs éléments. Si MultiSelect a la valeur False, il n'est pas possible de sélectionner à la fois plusieurs éléments de la boîte liste.

MultiSelect permet à l'utilisateur de sélectionner plusieurs éléments non contigus. Elle ne permet pas à l'utilisateur de sélectionner une séquence d'éléments en une seule opération comme le permet la propriété ExtendedSelect.

Si MultiSelect a la valeur True et que plusieurs éléments sont sélectionnés, la valeur de la propriété ItemIndex, indiquant l'indice de l'élément sélectionné (Selected), donne l'indice de l'élément sélectionné détenant la focalisation.

`ItemIndex`

type entier:
A l'exécution, utilisez la propriété ItemIndex pour sélectionner un élément. Affectez à la propriété ItemIndex l'indice de l'élément à sélectionner. La valeur de la propriété ItemIndex pour le premier élément de la liste est 0. Si aucun élément n'est sélectionné, la valeur est -1 (valeur par défaut) sauf si MultiSelect a la valeur True.

Si la propriété MultiSelect a la valeur True, l'utilisateur peut sélectionner plusieurs éléments de la boîte liste. Dans ce cas, la valeur de la propriété ItemIndex est l'indice de l'élément sélectionné qui détient la focalisation. Si MultiSelect a la valeur True, ItemIndex a par défaut la valeur 0.

`Items[i]`

type String:
Désigne le titre (chaîne de caractère) de l'élément i

Edit

Utilisez un objet TEdit pour placer dans une fiche un contrôle de saisie Windows standard. Les contrôles de saisie permettent à l'utilisateur de saisir du texte. Les contrôles permettent également d'afficher du texte.

Même pour afficher uniquement du texte, choisissez un contrôle de saisie pour permettre à l'utilisateur de sélectionner le texte et de le copier dans le Presse-Papiers. Choisissez un objet libellé si la possibilité de sélection qu'offre le contrôle de saisie n'est pas utile.

Les propriétés :

`CharCase`

Utilisez la propriété CharCase pour faire passer le contenu du contrôle de saisie en majuscules ou en minuscules. Les valeurs possibles de la propriété CharCase sont :

Valeur	Signification
`ecLowerCase`	Le texte est converti en minuscules.
`ecNormal`	Le texte apparaît tel quel, il n'y a pas de conversion.
`ecUpperCase`	Le texte est converti en majuscules.

Quand CharCase prend la valeur ecLowerCase ou ecUpperCase, les caractères sont convertis au fur et à mesure que l'utilisateur les entre dans le contrôle de saisie. L'affectation de la valeur ecLowerCase ou ecUpperCase à la propriété CharCase change réellement le texte, pas simplement son aspect. Toute information sur la

distinction minuscules/majuscules est perdue et n'est pas rétablie en affectant la valeur ecNormal à la propriété CharCase.

MaxLength
Utilisez la propriété MaxLength pour limiter le nombre de caractères pouvant être entrés dans le contrôle de saisie. Une valeur nulle indique qu'il n'y a pas de limite, définie par l'application, à la longueur..

Utilisez MaxLength pour limiter la longueur du texte d'un contrôle de saisie si le texte doit être copié dans un tampon de taille fixe.

Remarque : L'affectation d'une valeur à MaxLength ne tronque pas le texte existant, elle empêche simplement l'utilisateur d'ajouter du texte après avoir atteint la limite de MaxLength caractères.

Remarque : Même si MaxLength a la valeur 0, il peut y avoir des limitations du nombre de caractères pouvant être saisis dans le contrôle de saisie imposées par le système d'exploitation.

PasswoedChar
Utilisez la propriété PasswordChar pour créer un contrôle de saisie affichant un caractère spécial à la place de chaque caractère entré. Si PasswordChar a la valeur caractère nulle (caractère ANSI zéro), le contrôle de saisie affiche le texte normalement. Si PasswordChar contient tout autre caractère, le contrôle de saisie affiche le caractère spécifié par PasswordChar à la place de chaque caractère entré. PasswordChar n'affecte que l'aspect du contrôle de saisie. La valeur de la propriété Text contient bien les caractères effectivement saisis.

Text
Utilisez la propriété Text pour lire la valeur de Text pour le contrôle ou pour affecter une nouvelle chaîne à la valeur de Text. Par défaut, Text contient le nom du contrôle. Pour les contrôles boîte de saisie et mémo, la valeur de Text apparaît dans le contrôle. Pour les boîtes à options, la valeur de Text apparaît dans la partie contrôle de saisie de la boîte à options.

Remarque : Les contrôles affichant du texte utilisent soit la propriété Caption soit la propriété Text pour spécifier la valeur texte. La propriété utilisée dépend du type de contrôle. En général, Caption est utilisée pour le texte apparaissant comme un titre ou un libellé de fenêtre, alors que Text est utilisée pour le texte apparaissant sous forme de contenu d'un contrôle.

ComboBox

Un composant TComboBox est une boîte de saisie associée à une liste déroulante. Les utilisateurs peuvent sélectionner un élément de la liste ou entrer directement une valeur dans la boîte de saisie.

Propriétés :

`Items`

type TString:
Utilisez la propriété Items pour accéder à la liste d'éléments apparaissant dans la liste de la boîte à options.

`Text`

type String:
Lélément selectionné dans le ComboBox sera placé dans cette propriété

`Add`

(voir ListBox)


Memo & RichEdit

Memo
Utilisez TMemo pour placer dans une fiche un contrôle de saisie multiligne Windows standard. Les boîtes de saisie multilignes permettent à l'utilisateur de saisir plusieurs lignes de texte. Ces composants sont appropriés pour la représentation de données volumineuses.

RichEdit
Utilisez un objet TRichEdit pour placer un contrôle standard Windows éditeur de texte formaté dans une fiche. Les contrôles éditeur de texte formaté permettent à l'utilisateur de saisir du texte avec divers attributs de fonte et de mise en forme des paragraphes.

TRichEdit propose les propriétés et méthodes pour saisir et manipuler du texte formaté. Cependant TRichEdit ne propose aucun composant d'interface utilisateur pour permettre à l'utilisateur d'accéder à ses options de formatage du texte.

Propriétés :

`Lines`

type TString:
Utilisez la propriété Lines pour manipuler le texte d'un contrôle mémo ligne par ligne. Lines est un objet TStrings, il est donc possible d'utiliser les méthodes de TStrings avec Lines afin d'effectuer des manipulations comme compter les lignes de texte, ajouter de nouvelles lignes, supprimer des lignes ou remplacer les lignes par un autre texte.

Pour manipuler la totalité du texte d'un seul bloc, utilisez la propriété Text. Pour manipuler les lignes une par une, la propriété Lines est plus commode.

Exemple :
```
procedure TForm1.Button1Click(Sender: TObject);
begin
Showmessage('La 6ième ligne de votre texte est: '+Memo1.Lines[5]);
end;
```

Vous pouvez utiliser aussi **Add** pour ajouter des lignes:
```
Memo1.Lines.add('une phrase') ;
```

Uiliser aussi **count** pour connaître le nombre de lignes :
Exemple:
Cet exemple affiche le nombre de lignes dans un Memo en cliquant sur un bouton.
```
procedure TForm1.Button1Click(Sender: TObject);
begin
  showmessage(inttostr(memo1.lines.Count))
end;
```

StringGrid

Ajoutez un objet TStringGrid à une fiche pour proposer des données textuelles dans un format tabulaire. TStringGrid propose diverses propriétés pour contrôler l'aspect de la grille ainsi que des événements et méthodes tirant profit de la disposition tabulaire de la grille pour répondre aux actions de l'utilisateur.

TStringGrid introduit la possibilité d'associer un objet à chaque chaîne de la grille. Ces objets peuvent encapsuler des informations ou des comportements représentés par les chaînes proposées à l'utilisateur.

Si les chaînes à afficher dans la grille représentent les valeurs de champs des enregistrements d'un ensemble de données, utilisez TDBGrid.

Les propriétés	**Type**	**Définition**
ColCount	Integer	Nombre de colonnes
RowCount	Integer	Nombre de lignes
FixedColor	Integer	Couleur des colonnes fixes
FixedCols	Integer	Nombre des colonnes fixes
FixedRows	Integer	Nombre des lignes fixes
+*Options*		
goEditing	Boolean	Les utilisateurs peuvent éditer le contenu des cellules.
goTabs	Boolean	Les utilisateurs peuvent parcourir les cellules de la grille en utilisant Tab et Maj+Tab.
goAlwaysShowEditor	Boolean	La grille est verrouillée en mode modification. L'utilisateur n'a pas besoin d'utiliser Entrée ou F2 pour inverser EditorMode. Si Options n'inclut pas goEditing, goAlwaysShowEditor n'a aucun effet.

Exemple:
```
StringGrid1.ColCount := 5; (5 colonnes)
StringGrid1.RowCount := 6; (6 lignes)
```

Col & Row

Pour détecter la colonne et la ligne de la cellule sélectionnée
Exemple :
Cet exemple utilise une grille chaîne avec un libellé au-dessus sur une fiche. Quand l'utilisateur cliquer sur une cellule de la grille, l'emplacement du curseur s'affiche dans le libellé.

```
procedure TForm1.StringGrid1Click(Sender: TObject);
begin
  Label1.Caption := 'Le curseur se trouve dans la colonne ' +
                    IntToStr(StringGrid1.Col + 1) +
                    ', en ligne ' +
                    IntToStr(StringGrid1.Row + 1);
end;
```

Cells

`StringGrid1.Cells[i,j]`:représente la cellule qui se trouve à la colonne i, ligne j
Exemple : Dans l'évènement OnCreate de votre fiche (form1) taper les instructions:

```
procedure TForm1.FormCreate(Sender: TObject);
begin
    StringGrid1.Cells[1,0] := 'Nom';
    StringGrid1.Cells[2,0] := 'Prénom';
    StringGrid1.Cells[3,0] := 'Adresse';
end;
```

Chapitre

6

OnDragOver & OnDragDrop

OnDragOver Se produit quand l'utilisateur fait glisser un objet au-dessus d'un contrôle.
OnDragDrop Se produit quand l'utilisateur dépose un objet qu'il faisait glisser.

Description

OnDragOver

Utilisez un événement OnDragOver pour signaler que le contrôle autorise l'utilisateur à relâcher l'objet qu'il fait glisser.

Dans le gestionnaire d'événement OnDragOver, affectez la valeur False au paramètre Accept pour rejeter l'objet déplacé. Laissez Accept à la valeur True pour permettre à l'utilisateur de déposer l'objet sur le contrôle.

Pour modifier la forme du pointeur de la souris et indiquer que le contrôle autorise l'utilisateur à relâcher l'objet, modifiez la valeur de la propriété DragCursor du contrôle avant la fin du gestionnaire OnDragOver.

Source indique l'objet que l'utilisateur fait glisser, Sender est l'emplacement potentiel où sera déposé l'objet, et X et Y sont des coordonnées écran exprimées en pixels. Le paramètre State spécifie comment l'objet déplacé se déplace au-dessus du contrôle.

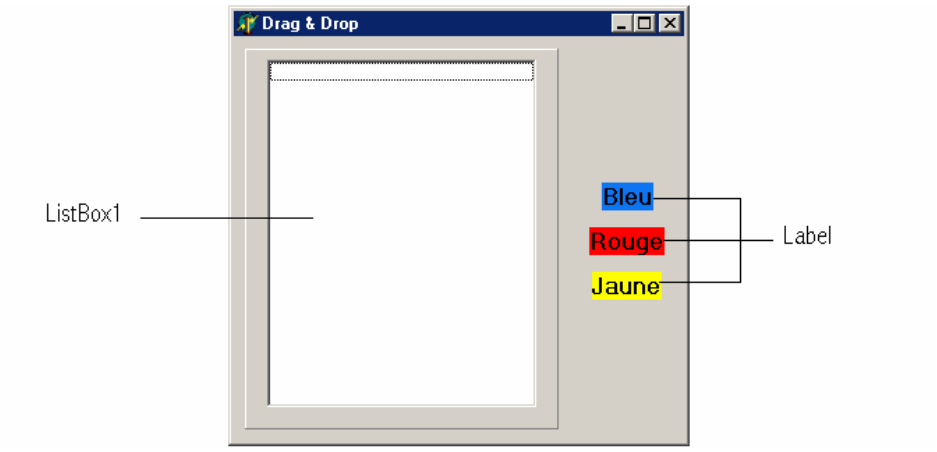
Remarque : Dans le gestionnaire d'événement OnDragOver, la valeur par défaut du paramètre Accept est True. Toutefois, si aucun gestionnaire d'événement n'est fourni, le contrôle rejette l'objet déplacé comme si le paramètre Accept avait la valeur False.

OnDragDrop

Utilisez le gestionnaire d'événement OnDragDrop pour écrire du code exécuté lorsque l'utilisateur relâche un objet. Le paramètre Source de l'événement OnDragDrop indique l'objet relâché, et Sender le contrôle sur lequel l'objet est relâché. Les paramètres X et Y indiquent les coordonnées de la souris au-dessus du contrôle.

Ce code provient d'une application qui contient une boîte liste (ListBox) et trois libellés (Label), chacun avec une couleur et fonte différentes. L'utilisateur peut sélectionner un libellé et le glisser vers une boîte liste. Lorsque le libellé est déplacé, les éléments de la boîte liste ont la fonte et la couleur de ce dernier.

REMARQUE : Initialiser la propriété ***DragMode*** de chaque libellé (Label) à ***dmAutomatic.***



Ce gestionnaire d'événement ***OnDragOver*** permet à la boîte liste d'accepter un libellé déplacé :

Cliquer deux fois sur l'événement ***OnDragOver*** de ListBox1, puis écrire l'instruction suivante :

```
procedure TForm1.ListBox1DragOver(Sender, Source: TObject; X, Y:
Integer; State: TDragState; var Accept: Boolean);
begin
  Accept := Source is TLabel;
end;
```

Ce gestionnaire d'événement ***OnDragDrop*** implémente le comportement d'un objet "déplacé".

Cliquer deux fois sur l'événement ***OnDragDrop*** de ListBox1, puis écrire les instructions suivantes :

```
procedure TForm1.ListBox1DragDrop(Sender, Source: TObject; X, Y:
Integer);
begin
  if (Sender is TListBox) and (Source is TLabel) then
  begin
    with Sender as TListBox do
    begin
      Font := (Source as TLabel).Font;
      Color := (Source as TLabel).Color;
    end;
  end;
end;
```

Chapitre

7

Objets pour manipuler une Base de Données

AccèsBD

DataSource

TDataSource sert d'interface entre un composant ensemble de données et les contrôles orientés données d'une fiche.
La classe TDataSource sert de canal entre un ensemble de données et les contrôles orientés données d'une fiche permettant ainsi l'affichage, les déplacements, et la modification des données de l'ensemble de données sous-jacent.
Pour que les données d'un ensemble de données puissent être affichées et manipulées dans des contrôles orientés données, l'ensemble de données doit être associé à un composant source de données. De même, chaque contrôle orienté données doit être associé à un composant source de données pour pouvoir recevoir et manipuler les données.
Les composants source de données servent également à relier les ensembles de données dans une relation maître-détail.

Propriété :
`DataSet` : Permet de relier le DataSource à une Table

Table

TTable encapsule une table de base de données.
La classe TTable permet d'accéder aux données d'une table de base de données en utilisant le moteur de bases de données Borland (BDE). TTable donne un accès direct à chaque enregistrement et à chaque champ de la table d'une base de données sous-jacente, que ce soit une base de données Paradox, dBASE, Access, FoxPro, une base de données ODBC ou une base de données SQL dans un serveur distant comme InterBase, Oracle, Sybase, MS-SQL Server, Informix ou DB2. Un

composant table peut également travailler sur un sous-ensemble d'enregistrements d'une table en utilisant des intervalles et des filtres.
Au moment de la conception, vous pouvez créer, supprimer, actualiser ou renommer la table base de données connectée à TTable en cliquant avec le bouton droit sur TTable et en utilisant le menu surgissant.

Propriétés :

`Active` :

ouvre automatiquement la table

`DatabaseName` :

La propriété DatabaseName spécifie le nom de la base de données à utiliser avec un composant base de données. Si DatabaseName contient un nom d'alias existant du moteur de bases de données Borland (BDE), il n'est pas nécessaire de définir la valeur des propriétés AliasName et DriverName. Si DatabaseName ne correspond pas à un alias BDE existant, l'application doit spécifier un nom d'alias existant dans la propriété AliasName en plus de DatabaseName, ou elle doit définir la valeur des propriétés DriverName et Params.
DatabaseName peut contenir un chemin d'accès complet pour se connecter à une base de données Paradox ou dBASE.

`ReadOnly` :

possibilité ou non de modification de la table

`TableName` :

La propriété TableName permet de spécifier le nom de la table de base de données encapsulée par ce composant. Pour affecter à TableName une valeur significative, il faut avoir déjà défini la propriété DatabaseName. A la conception, si DatabaseName est définie, il est possible de sélectionner un nom de table valide dans la liste déroulante de la propriété TableName dans l'inspecteur d'objets.

Remarque : Pour définir TableName, il faut que la propriété Active ait la valeur False.

`Eof` : indique c'est la fin de la table a été atteinte

Database

TDatabase permet, dans une application de base de données, de contrôler précisément la connexion avec une base de données basée sur le BDE.
Utilisez un objet TDatabase quand une application de base de données basée sur le BDE nécessite l'une des options suivantes sur la connexion avec une base de données :
Connexion persistante avec la base de données.
Accès personnalisé à un serveur de bases de données.
Contrôle des transactions.
Alias BDE spécifique à l'application

La classe TDatabase est particulièrement utile car elle permet de contrôler le traitement des transactions par le BDE lors d'une connexion avec un serveur SQL distant.
Remarque : La déclaration explicite d'un composant TDatabase pour chaque connexion de base de données n'est pas nécessaire pour une application n'ayant pas besoin de contrôler explicitement une connexion. Si un composant TDatabase n'est pas explicitement déclaré et instancié dans une connexion de base de données, un composant base de données temporaire est créé à l'exécution, utilisant des valeurs par défaut pour les propriétés.

`AliasName`	Choisissez un Alias
`DatabaseName`	Donnez un nom
`LoginPrompt`	Initialisez à False

Query

TQuery encapsule un ensemble de données et un ensemble de résultats basés sur une instruction SQL.
La classe TQuery permet d'accéder à une ou plusieurs tables d'une base de données en utilisant des instructions SQL. Les composants requête peuvent être utilisés avec des serveurs de bases de données distants comme Sybase, SQL Server, Oracle, Informix, DB2, et InterBase, et avec des tables locales Paradox, InterBase, dBASE, Access et FoxPro, ou avec des bases de données ODBC.
Les composants requête sont pratiques, car ils peuvent :
Accéder à plusieurs tables à la fois (ce qu'on appelle en SQL une «jointure»).
Accéder automatiquement à un sous-ensemble des lignes et colonnes de leurs tables sous-jacentes au lieu de renvoyer systématiquement toutes les lignes et toutes les colonnes.

Remarque :
TQuery joue un rôle encore plus important dans le développement d'applications de base de données évolutives. S'il y a le moindre risque qu'une application conçue pour fonctionner avec des bases de données locales soit ultérieurement déployée sur un serveur de bases de données, utilisez des composants TQuery dès le départ pour faciliter la migration plus tard.

ControleBD

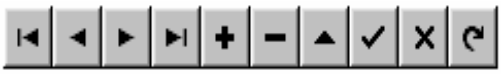
DBGrid

TDBGrid affiche et manipule les enregistrements d'un ensemble de données dans une grille tabulaire.
Placez un objet TDBGrid dans une fiche afin d'afficher et d'éditer les enregistrements d'une table de base de données ou d'une requête. Une application peut utiliser une

grille de données pour insérer, supprimer ou modifier les données de la base de données ou simplement pour les afficher.
A l'exécution, les utilisateurs peuvent utiliser le navigateur de base de données (TDBNavigator) pour se déplacer dans les données de la grille, et pour insérer, supprimer ou modifier les données. Les modifications effectuées dans la grille de données ne sont pas transmises aux données sous-jacentes tant que l'utilisateur ne s'est pas déplacé sur un enregistrement différent ou n'a pas quitté l'application.
TDBGrid implémente les comportements génériques introduits dans TCustomDBGrid. TDBGrid publie de nombreuses propriétés héritées de TCustomDBGrid, mais n'introduit aucun comportement nouveau.

Proprété :
`DataSource` Donnez le DataSource correspondant

DBNavigator



Le navigateur de base de données (TDBNavigator) est utilisé pour se déplacer dans un ensemble de données et effectuer des actions sur les données (par exemple, insérer un nouvel enregistrement ou expédier un enregistrement).
Utilisez un navigateur de base de données dans des fiches contenant des contrôles orientés données comme TDBGrid ou TDBEdit. TDBNavigator permet à l'utilisateur de contrôler l'ensemble de données en modification ou en consultation.
Quand l'utilisateur choisit un des boutons du navigateur, l'action correspondante est effectuée sur l'ensemble de données auquel est lié le navigateur. Si par exemple, l'utilisateur clique sur le bouton Insérer, un nouvel enregistrement est inséré dans l'ensemble de données.

TDBNavigator peut afficher les boutons suivants :

Bouton	**Rôle**
Premier	Le premier enregistrement de l'ensemble de données devient l'enregistrement en cours, les boutons Premier et Précédent sont désactivés et les boutons Suivant et Dernier sont activés.
Précédent	L'enregistrement précédent devient l'enregistrement en cours, les boutons Dernier et Suivant sont activés.
Suivant	L'enregistrement suivant devient l'enregistrement en cours et les boutons Premier et Précédent sont activés.
Dernier	Le dernier enregistrement de l'ensemble de données devient l'enregistrement en cours, les boutons Dernier et Suivant sont désactivés et les boutons Premier et Précédent sont activés.
Insérer	Insère un nouvel enregistrement qui précède l'enregistrement en cours et bascule l'ensemble de données en mode insertion et modification.
Supprimer	Supprime l'enregistrement en cours et l'enregistrement suivant devient l'enregistrement en cours.
Modifier	Bascule l'ensemble de données en mode modification pour que l'enregistrement en cours puisse être modifié.
Emettre	Ecrit les modifications de l'enregistrement en cours dans la base de données.

Annuler	Annule les modifications dans l'enregistrement en cours et restitue l'état de l'affichage de l'enregistrement tel qu'il était avant la modification, désactive les modes insertion et modification s'ils sont actifs.
Rafraîchir	Actualise les données du tampon de l'ensemble de données associé. Pour les composants TQuery, ce bouton est désactivé sauf si la propriété RequestLive a la valeur True.

Proprété :
`DataSource` Donnez le DataSource correspondant

DBText

TDBText représente un contrôle orienté données qui affiche la valeur d'un champ dans une fiche.
Utilisez TDBText pour afficher dans une fiche le contenu d'un champ de l'enregistrement en cours d'un ensemble de données. Les valeurs de champ affichées par des contrôles texte de base de données ne peuvent être modifiées par l'utilisateur dans le contrôle texte. Pour permettre à l'utilisateur de modifier le contenu du champ, utilisez TDBEdit ou TDBMemo.
Si votre application n'a pas besoin de l'orientation données de la classe TDBText, utilisez plutôt le composant texte (TLabel) ou le composant texte statique (TStaticText) plus économes en ressources système.

Proprété :
`DataSource` Donnez le DataSource correspondant

DBEdit

TDBEdit représente un contrôle de saisie monoligne pouvant afficher et modifier les valeurs d'un champ d'un ensemble de données.
Utilisez TDBEdit pour permettre aux utilisateurs de modifier un champ de base de données. TDBEdit utilise la propriété Text pour représenter le contenu du champ.
TDBEdit ne permet la saisie que d'une seule ligne de texte. Si un champ peut contenir des données volumineuses nécessitant plusieurs lignes, utilisez un objet TDBMemo.
Si votre application n'a pas besoin de l'orientation données de la classe TDBEdit, utilisez plutôt un contrôle de saisie (TEdit) ou un contrôle de saisie masqué (TMaskEdit) plus économe en ressources système.

Pour fournir un masque restreignant la saisie et contrôlant le format d'affichage des données, utilisez les propriétés relatives au masque de TField et ses descendants. Ces propriétés comprennent : TField.::EditMask, TDateTimeField.::DisplayFormat et TNumericField.::DisplayFormat. La propriété devant être utilisée est indiquée dans le type du champ ainsi que le descendant TField correspondant à ce type.

Propriété :
`DataSource` Donnez le DataSource correspondant DataField :

DBMemo

TDBMemo représente un contrôle de saisie multiligne pouvant afficher et modifier un champ d'un ensemble de données.
Utilisez TDBMemo pour permettre à l'utilisateur de modifier ou de consulter un champ contenant du texte, même volumineux. TDBMemo utilise la propriété Text pour représenter le contenu du champ.
TDBMemo gère plusieurs lignes de texte. Cet objet est donc adapté aux champs alphanumériques longs et aux champs texte BLOB (objet binaire volumineux). Pour les champs alphanumériques plus courts, mieux vaut utiliser un composant TDBEdit.
Si votre application n'a pas besoin de l'orientation données de la classe TDBMemo, utilisez plutôt le contrôle mémo (TMemo) plus économe en ressources système.

Propriété :
`DataSource` Donnez le DataSource correspondant
`DataField` Donner le champ correspondant

DBListBox

TDBListBox est une boîte liste orientée données qui permet à l'utilisateur de changer la valeur du champ de l'enregistrement en cours d'un ensemble de données en sélectionnant un élément dans une liste.
Utilisez TDBListBox pour ajouter à une fiche une boîte liste permettant aux utilisateurs de changer la valeur d'un champ à partir d'un ensemble de choix prédéfinis.
Si l'application n'a pas besoin des fonctionnalités orientées données de TDBListBox, utilisez plutôt une boîte liste (TListBox) pour économiser des ressources système.
Les créateurs de composants souhaitant créer des objets boîte liste personnalisés doivent utiliser TCustomListBox comme classe de base.

Propriété :
`Items` Pour ajouter les éléments de la liste.

DBComboBox

TDBComboBox représente un contrôle boîte à options orienté données.
Utilisez TDBComboBox pour permettre à l'utilisateur de changer la valeur d'un champ de l'enregistrement en cours dans un ensemble de données, en sélectionnant un élément dans une liste ou en tapant des informations dans la boîte de saisie du contrôle. L'élément sélectionné ou le texte tapé devient la nouvelle valeur du champ si la propriété ReadOnly de la boîte à options de base de données a la valeur False. La boîte à options peut être personnalisée pour autoriser ou interdire la saisie dans la boîte de saisie du contrôle, pour afficher la liste sous forme déroulante ou de manière permanente, pour trier les éléments de la liste, etc.

Proprété :
`Items` Pour ajouter les éléments de la liste.

DBLookupListBox

TDBLookupListBox propose une liste d'éléments de référence pour renseigner les champs qui attendent des valeurs provenant d'autres ensembles de données.
Utilisez TDBLookupListBox pour proposer aux utilisateurs une liste d'éléments de référence permettant de définir la valeur d'un champ en utilisant les valeurs d'un champ d'un autre ensemble de données. Les boîtes listes de référence affichent habituellement des valeurs décrivant la valeur réelle du champ.
Si TDBLookupListBox est lié à un composant champ de référence, il bénéficie automatiquement de la relation entre la valeur du champ et les valeurs de référence dans l'ensemble de données de référence à partir du composant champ. La relation entre les valeurs de champ et les valeurs correspondantes dans l'ensemble de données de référence peut aussi être définie explicitement en utilisant les propriétés de la boîte liste de référence quand la boîte liste n'est pas liée à un composant champ de référence.

Proprété :

`DataSource`	Donnez le DataSource correspondant
`DataField`	Donner le champ correspondant
`ListSource`	Identifie une source de données pour les données affichées dans le contrôle de référence.
`ListField`	dentifie les champs dont les valeurs sont affichées dans le contrôle de référence.
`KeyField`	Identifie le champ de l'ensemble de données ListSource qui doit correspondre à la valeur du champ DataField.

DBLookupComboBox

TDBLookupComboBox représente une boîte à options qui identifie un ensemble de valeurs de champ d'un ensemble de données à l'aide d'un ensemble de valeurs correspondantes d'un autre ensemble de données.
Utilisez TDBLookupComboBox pour fournir à l'utilisateur une liste déroulante d'éléments de références pour l'initialisation de champs nécessitant des données d'un autre ensemble de données.
Si TDBLookupComboBox est lié à un composant champ de références, il utilise automatiquement la relation entre la valeur du champ et les valeurs de références dans l'ensemble de données de références du composant champ. La relation entre les valeurs du champ et les valeurs correspondantes dans l'ensemble de données de références peut aussi être définie de manière explicite en utilisant les propriétés de la boîte à options de références si la boîte à options n'est pas liée à un composant champ de références.

Proprété :

DataSource	Donnez le DataSource correspondant
DataField	Donner le champ correspondant
ListSource	Identifie une source de données pour les données affichées dans le contrôle de référence.
ListField	dentifie les champs dont les valeurs sont affichées dans le contrôle de référence.
KeyField	Identifie le champ de l'ensemble de données ListSource qui doit correspondre à la valeur du champ DataField.

QReport

QuickRep

Utilisez TQuickRep pour créer un état connecté à un ensemble de données.
TQuickRep est le composant d'état le plus répandu. Pour plus d'informations sur la création des états, reportez-vous au guide de l'utilisateur.
Propriétés

Dataset	voir la définition de DataSet
RecordCount	RecordCount essaie de renvoyer le nombre d'enregistrements d'un ensemble de données maître. QuickReport utilise cette information pour maintenir la barre de progression pendant la génération d'état. Certaines bases de données ne peuvent renvoyer le compteur d'enregistrements. Pour ces bases de données, la barre de progression ne sera pas affichée.
RecordNumber	RecordNumber renvoie le numéro de l'enregistrement en cours de traitement par l'état. Le premier enregistrement / ligne est 0.

Pour afficher la page QuickRep en mode exécution, utilisez la méthode `Preview` :

```
QuickRep1.Preview ;
```

QRLabel

L'emploi du composant TQRLabel ressemble à un Tlabel ordinaire, valable seulement pour les états. Utilisez ce composant pour imprimer du texte statique sur un état en entrant du texte dans la propriété Caption.
Vous pouvez aussi modifier le texte à imprimer dans l'événement OnPrint, ce qui rend facile l'impression des calculs ou d'autres informations.

Propriétés

`DataSet`	Choisir une table
`DataField`	Choisir un champ de la table

QRDBText

Utilisez le composant TQRDBText pour imprimer tout champ texte d'une table, à l'exception des champs mémo au format RichText. TQRDBText peut imprimer des champs alphanumériques, des champs à virgule flottante, des champs date et heure et des champs mémo.
TQRDBText peut s'étendre verticalement pour s'adapter au texte, si celui-ci excède la taille originale. La propriété AutoStretch doit être définie à True pour permettre l'étirement. Un composant peut s'étendre sur plusieurs pages.
Pour formater les sorties, utilisez la propriété Mask.

Propriétés

`DataSet`	Choisir une table
`DataField`	Choisir un champ de la table
`Mask`	Utilisez la propriété Mask pour formater le résultat du composant TQRDBText. Si aucun masque n'est spécifié, QuickReport utilise le formatage par défaut pour le champ. Pour plus d'informations sur le formatage d'un champ numérique, reportez-vous à la rubrique FormatFloat de l'aide VCL.

QRExpr

Utilisez TQRExpr pour calculer une expression durant la génération d'un état. Pour plus d'informations sur l'utilisation des expressions, reportez-vous au guide de l'utilisateur.
Le double-clic sur cette expression dans l'inspecteur d'objets affiche un constructeur d'expressions.
Dans la propriété Expression, entrez l'expression à évaluer.
Le changement d'expression n'est pas autorisé durant la génération d'un état.

Propriétés

`Expression` Dans la propriété Expression, entrez l'expression à évaluer. Le changement d'expression n'est pas autorisé durant la génération d'un état.

`Value` Utilisez la propriété Value pour accéder à la valeur de l'expression. Elle renvoie un enregistrement variant avec le résultat d'un calcul.

Méthodes

`Reset` Appelez Reset dans tout gestionnaire d'événement durant la génération d'un état pour la valeur d'une expression à réinitialiser à 0. Cela est utile lorsque vous souhaitez un contrôle manuel sur la réinitialisation des calculs.d'expressions récapitulatives

QRSysData

Utilisez TQRSysData pour imprimer des informations système, comme le titre de l'état et le numéro de page en cours. Dans la propriété Data, sélectionnez les données à imprimer. Dans la propriété Text, mettez le texte qui précède.

Propriété :

`Data` Utilisez la propriété Data pour sélectionner l'élément de données à imprimer par le composant TQRSysData.

Chapitre

8

Comment effectuer une recherche dans une Table ?

Pour effectuer une recherche dans une table on utilise les méthodes **`FindKey`**, et **`FindNearest`**

Exemples :

`FindKey` est utilisé pour effectuer une recherche dont le(s) champ(s) correspondant à l'index sont strictement égaux à la (aux) valeur(s) passée(s) en paramètre.

```
Table1.IndexFieldNames :='nom du champ dans la table' ;
Table1.FindKey(['chaîne de caractères']) ;
```

Du fait que la méthode `FindKey` est une fonction booléenne, on peut donc remplacer l'instruction `Table1.FindKey(['chaîne de caractères'])` par:

```
if Table1.FindKey(['chaîne de caractères']) then …     //si l'enregistrement est trouvé
else …                                                  //si l'enregistrement n'est pas trouvé
```

`FindNearest` est utilisé pour effectuer une recherche dont le(s) champ(s) correspondant à l'index sont proches de la (des) valeur(s) passée(s) en paramètre.

```
Table1.IndexFieldNames :='nom du champ dans la table' ;
Table1.FindNearest(['chaîne de caractères']) ;
```

Remarque :

L'instruction `Table1.IndexFieldNames` n'est pas obligatoire si le recherche se fait sur le champ clé de la table, il est donc évident de la supprimer.