Editeur de recherche et liste hiérarchique

Par
Jean-Luc PELLERIN, Formation 4D
Note technique 4D-200006-18-FR
Version 1

Date 1 Juin 2000

Résumé

Cette note technique vous a présenté une autre manière de réaliser un éditeur de recherche à l'aide d'une liste hiérarchique simple. L'objectif étant de comprendre le fonctionnement d'une liste hiérarchique simple.

4D Notes techniques

Introduction

L'éditeur de recherche standard de recherche présente de nombreuses fonctionnalités, mais il a le défaut de proposer une interface peut conviviale.
La maîtrise des listes hiérarchiques permet de proposer aux utilisateurs des outils de recherche plus adaptés.
L'objectif est de comprendre le fonctionnement d'une liste hiérarchique simple.

Principe de la base exemple

Le principe est de présenter sous forme de listes hiérarchiques des valeurs hiérarchisées. Un simple clic sur un élément permet de faire la sélection correspondante.

La base exemple

La base exemple présente une liste d'entreprises contenant un champ région et un champ département.

La programmation

La phase d'initialisation

Dans un premier temps, il faut initialiser la variable qui contient la référence à la liste dans la méthode base

```
``Méthode base sur ouverture
◊LH:=0
```

À l'ouverture de l'interface, on initialise la liste hiérarchique sur la méthode objet du bouton initialiser.

1/ Nous supprimons la liste hiérarchique si elle existe

```
`  Liste_H ("Initialiser_Interface")
   Liste_H ("Supprimer_Liste")`Suppression de l'ancienne liste éventuelle

`  Liste_H("Supprimer_Liste")
   Si (Liste existante (◊LH))
     SUPPRIMER LISTE (◊LH ;*)
   Fin de si
```

2/ À partir de la sélection, on créé un tableau local afin de récupérer la liste des régions

```
TOUT  SELECTIONNER ([Clients])
  `La sélection comprend tous les enregistrements
VALEURS  DISTINCTES ([Clients] Région ;$T_Region)
  `Le tableau contient la liste de toutes les régions
REDUIRE  SELECTION ([Clients] ;0)
```

3/ Nous créons la liste hiérarchique puis nous ajoutons de nouveau éléments à la liste pour présenter la liste des régions. Chaque élément de la liste sera affecté par un élément de tableau et valeur une valeur unique délivrée par la routine Hasard.

```
◊LH :=Nouvelle  liste
Boucle ($i ;1 ;Taille  tableau ($T_Region))
 AJOUTER  A  LISTE (◊LH ;$T_Region {$i}; Hasard)
Fin  de  boucle
```

La phase d'utilisation

Un clic sur un élément de la liste hiérarchique devra provoquer les actions suivantes :

1/ Connaître le niveau dans la liste, en effet on ne fera pas la même chose si l'élément est au premier niveau ou au second.
Pour connaître le niveau on va boucler à la recherche de l 'élément père. Lorsque la routine Element parent retourne la valeur zéro, nous sommes en présence d'un élément de niveau un.

```
``  Liste_H("Liste_Niveau")
$Element_Sel :=Element  selectionne (◊LH)
INFORMATION  ELEMENT (◊LH ;$Element_Sel ;$Element_Ref ;$Tx)
$Element_Ref :=Element  parent (◊LH ;$Element_Ref)
$0:=1
Tant  que ($Element_Ref>0)
  $Element_Ref :=Element  parent (◊LH ;$Element_Ref)
  $0:=$0+1
Fin  tant  que
```

Après avoir obtenu le niveau de l'élément, si nous sommes au premier niveau de la liste, nous allons rechercher les entreprises du département concerné, puis construire une sous-liste contenant ces éléments, à la condition qu'elle n'existe pas.
Cette sous-liste sera associée à l'élément du niveau supérieur à l'aide de la routine CHANGER ELEMENT.
Il est préférable de demander explicitement le redessinement de la liste hiérarchique avec la routine REDESSINER LISTE.

Lorsque l'on se trouve en présence d'un élément de niveau 2, on fait une recherche des entreprises du département.

```
Au cas ou

    : ($Niveau=1) `Faire liste des Dpt
      $Element_Sel :=Element selectionne (◊LH)
     INFORMATION ELEMENT (◊LH ;$Element_Sel ;$Element_Ref ;$Tx ;$Sous_Liste)
     CHERCHER ([Clients] ;[Clients] Région=$Tx)

     S i ($Sous_Liste=0)
      VALEURS DISTINCTES ([Clients] Département ;$T_Dpt)
       $Sous_Liste :=Nouvelle liste
      Boucle ($i ;1 ;Taille tableau ($T_Dpt))
       AJOUTER A LISTE ($Sous_Liste ;$T_Dpt {$i};Hasard)
      Fin de boucle
      CHANGER ELEMENT (◊LH ;$Element_Ref ;$Tx ;$Element_Ref ;$Sous_Liste ;Vrai)
     Fin de si
     REDESSINER LISTE (◊LH)

    : ($Niveau=2) `Faire liste des entreprisses
      $Element_Sel :=Element selectionne (◊LH)
     INFORMATION ELEMENT (◊LH ;$Element_Sel ;$Element_Ref ;$Tx ;$Sous_Liste)
     CHERCHER ([Clients] ;[Clients] Département=$Tx)


  Fin de cas
```

La phase de sortie

À la sortie du formulaire, on ne doit pas oublier de supprimer la liste hiérarchique de la mémoire vive. En effet, ce qui est contenu dans le variable retourné par nouvelle liste c'est la référence à la liste.

Lorsque le process est tué, la variable process disparaîtra avec la référence de la liste hiérarchique mais non la liste elle-même. La liste hiérarchique restera en mémoire vive et sera inaccessible.

La suppression se fera avec la routine SUPPRIMER LISTE ajoutée d'une* afin que toutes les sous-listes soient aussi effacées de la mémoire.

```
``  Liste_H("Supprimer_Liste")

   S i (Liste existante (◊LH))
     SUPPRIMER LISTE (◊LH ;*)

   Fin de si
```

La variable ◊LH contenant la référence à la liste hiérarchique est variable interprocess afin que cette référence n'existe qu'une seule fois dans la mémoire vive.

L'utilisation d'une variable process aurait conduit à la duplication de cette variable dans tous les process.