

4D v11 SQL

*Autoformation
Windows®/Mac OS®*



4D v11 SQL

Autoformation

Copyright© 1985 - 2007 4D SAS
Tous droits réservés.

Les informations contenues dans ce manuel peuvent faire l'objet de modifications sans préavis et ne sauraient en aucune manière engager 4D SAS. La fourniture du logiciel décrit dans ce manuel est régie par un octroi de licence dont les termes sont précisés par ailleurs dans la licence électronique figurant sur le support du Logiciel et de la Documentation y afférente. Le logiciel et sa Documentation ne peuvent être utilisés, copiés ou reproduits sur quelque support que ce soit et de quelque manière que ce soit, que conformément aux termes de cette licence.

Aucune partie de ce manuel ne peut être reproduite ou recopiée de quelque manière que ce soit, électronique ou mécanique, y compris par photocopie, enregistrement, archivage ou tout autre procédé de stockage, de traitement et de récupération d'informations, pour d'autres buts que l'usage personnel de l'acheteur, et ce exclusivement aux conditions contractuelles, sans la permission explicite de 4D SAS.

4D, 4D Draw, 4D Write, 4D View, 4D Server ainsi que les logos 4D et 4D sont des marques enregistrées de 4D SAS.

Windows, Windows Vista, Windows XP et Microsoft sont des marques enregistrées de Microsoft Corporation.

Apple, Macintosh, QuickTime, Mac OS sont des marques enregistrées ou des noms commerciaux de Apple Computer, Inc.

Mac2Win Software est un produit de Altura Software, Inc.

ICU Copyright © 1995-2007 International Business Machines Corporation and others. All rights reserved.

Ce produit inclut un programme développé par Apache Software Foundation (<http://www.apache.org/>). 4D utilise des logiciels de cryptographie écrits par Eric Young (ey@cryptsoft.com), ainsi que des logiciels écrits par Tim Hudson (tjh@cryptsoft.com).

Correcteur orthographique, © Copyright SYNAPSE Développement, Toulouse, France, 1994-2007.

ACROBAT © Copyright 1987-2007, Secret Commercial Adobe Systems Inc. Tous droits réservés. ACROBAT est une marque enregistrée d'Adobe Systems Inc.

Tous les autres noms de produits ou appellations sont des marques déposées ou des noms commerciaux appartenant à leurs propriétaires respectifs.

Sommaire

Bienvenue	9	
Principes fondamentaux	11	
Présentation de l'application Exemple	13	
Chapitre 1	Premiers pas dans 4D	15
	Mise en oeuvre	15
	Découvrons l'interface	15
	Exercice	19
	Commentaires	20
Chapitre 2	Formulaires et boutons	23
	Mise en oeuvre	23
	Exercice	28
	Commentaires	28
Chapitre 3	Affichage au démarrage	29
	Mise en oeuvre	29
	Exercice	33
	Commentaires	33
Chapitre 4	Barres de menus	35
	Mise en oeuvre	35
	Exercice	38
	Commentaires	38
Chapitre 5	Page 0 et page 1	39
	Mise en oeuvre	39
	Exercice	44
	Commentaires	45

Chapitre 6	Pages de formulaire et navigation47
	Mise en oeuvre	47
	Exercice	49
	Commentaires	49
Chapitre 7	Tables et champs51
	Mise en oeuvre	51
	Exercice	54
	Commentaires	54
Chapitre 8	Saisie et modification d'enregistrements57
	Mise en oeuvre	57
	Exercice	61
	Commentaires	61
Chapitre 9	Liens63
	Mise en oeuvre	63
	Exercice	68
	Commentaires	68
Chapitre 10	Saisie et suppression69
	Mise en oeuvre	69
	Exercice	72
	Commentaires	73
Chapitre 11	Importer75
	Mise en oeuvre	75
	Exercice	77
	Commentaires	78
Chapitre 12	Formulaires de sortie79
	Mise en oeuvre	79
	Exercice	85
	Commentaires	85
Chapitre 13	Recherches et tris87
	Mise en oeuvre	87
	Trier	90

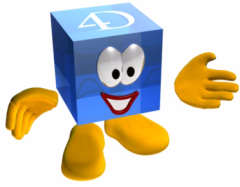
	Exercice	91
	Commentaires	92
Chapitre 14	Etats rapides et export	93
	Mise en oeuvre	93
	Exercice	101
	Commentaires	101
Chapitre 15	Recherche et tri par formule, appliquer une formule	103
	Mise en oeuvre	103
	Exercice	108
	Commentaires	109
Chapitre 16	Héritage de formulaires.	111
	Mise en oeuvre	111
	Exercice	114
	Commentaires	114
Chapitre 17	Formulaires impression	115
	Mise en oeuvre	115
	Exercice	118
	Commentaires	119
Chapitre 18	Manipuler les objets sur les formulaires . . .	121
	Mise en oeuvre	121
	Exercice	126
	Commentaires	126
Chapitre 19	Propriétés des objets	127
	Mise en oeuvre	127
	Exercice	131
	Commentaires	131
Chapitre 20	Calculs et formules	133
	Mise en oeuvre	133
	Exercice	135
	Commentaires	135

Chapitre 21	Présentation des variables	137
	Mise en oeuvre	137
	Exercice	140
	Commentaires	140
Chapitre 22	Mode trace et débogage	141
	Mise en oeuvre	141
	Exercice	147
	Commentaires	147
Chapitre 23	Programmation générique (sans pointeurs) .	149
	Mise en oeuvre	149
	Exercice	151
	Commentaires	152
Chapitre 24	Pointeurs.	155
	Mise en oeuvre	155
	Exercices	157
	Commentaires	158
Chapitre 25	Evénements	159
	Mise en oeuvre	159
	Exercice	163
	Commentaires	165
Chapitre 26	Tableaux, pop up, listbox	167
	Mise en oeuvre	167
	Exercice	173
	Commentaires	173
Chapitre 27	4D Write	175
	Mise en oeuvre	175
	Exercice	179
	Commentaires	179
Chapitre 28	Fenêtre et navigation	181
	Mise en oeuvre	181
	Exercice	186
	Commentaires	188

Chapitre 29	4D Internet Commands.	189
	Mise en oeuvre189
	Exercice192
	Commentaires192
Chapitre 30	Mots de passe et groupes.	193
	Mise en oeuvre193
	Exercice196
	Commentaires197
Chapitre 31	Triggers	199
	Mise en oeuvre200
	Exercice201
	Commentaires201
Chapitre 32	Sélection courante	203
	Mise en oeuvre204
	Exercice210
	Commentaires211
Chapitre 33	Ensembles et sélections temporaires	213
	Mise en oeuvre213
	Sélections temporaires.215
	Exercice216
	Commentaires217
Chapitre 34	Process	219
	Mise en oeuvre220
	Exercice222
	Commentaires223
Chapitre 35	SQL	225
	Mise en oeuvre225
	Exercice229
	Commentaires229

Chapitre 36	BLOBS	231
	Mise en oeuvre	232
	Exercice	235
	Commentaires	235
Chapitre 37	Corrigés	237
	Exercice : "Formulaires et boutons", page 28	237
	Exercice : "Affichage au démarrage", page 33	237
	Exercice : "Pages de formulaire et navigation", page 49	238
	Exercice : "Tables et champs", page 54	238
	Exercice : "Saisie et modification d'enregistrements", page 61	239
	Exercice : "Liens", page 68	239
	Exercice : "Saisie et suppression", page 72	239
	Exercice : "Importer", page 77	239
	Exercice : "Formulaires de sortie", page 85	240
	Exercice : "Recherches et tris", page 91	240
	Exercice : "Etats rapides et export", page 101	241
	Exercice : "Recherche et tri par formule, appliquer une formule", page 108	245
	Exercice : "Héritage de formulaires", page 114	246
	Exercice : "Propriétés des objets", page 131	246
	Exercice : "Calculs et formules", page 135	247
	Exercice : "Présentation des variables", page 140.	248
	Exercice : "Programmation générique (sans pointeurs)", page 151	249
	Exercice : "Pointeurs", page 157	250
	Exercice : "Événements", page 163	252
	Exercice : "Tableaux, pop up, listbox", page 173	253
	Exercice : "4D Write", page 179	253
	Exercice : "Fenêtre et navigation", page 186	255
	Exercice : "4D Internet Commands", page 192	256
	Exercice : "Mots de passe et groupes", page 196	257
	Exercice : "Triggers", page 201	258
	Exercice : "Sélection courante", page 210	259
	Exercice : "Ensembles et sélections temporaires", page 216	261
	Exercice : "Process", page 222	261
	Exercice : "SQL", page 229	264
	Exercice : "BLOBS", page 235	266
Conclusion		269
	Pour aller plus loin	270

Bienvenue

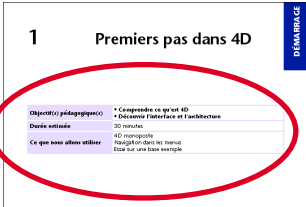
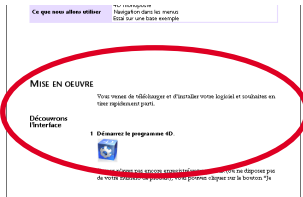



Bonjour et bienvenue dans votre nouvel environnement 4D.

Je suis *Prof4D*, votre assistant personnel, et tout au long de ce guide nous allons travailler ensemble.

Je vous propose de faire un rapide tour d'horizon des possibilités de 4D. Bien sûr, je ne serai pas exhaustif, mais à la fin de ce manuel, vous connaîtrez les principales étapes pour réaliser des bases de données fonctionnelles, ergonomiques et efficaces...

Chaque chapitre est composé de différents points :

<p>1) Tableau de présentation</p> 	<p>Un tableau de présentation de la leçon contenant :</p> <ul style="list-style-type: none"> - le ou les objectif(s) à atteindre, - la durée estimée de l'exercice, - la liste des éléments utilisés pendant l'exercice.
<p>2) Mise en oeuvre</p> 	<p>Les étapes et instructions pour la réalisation de la leçon</p>
<p>3) Exercices</p> 	<p>Des exercices supplémentaires qui prolongent la leçon et vérifient que vous avez assimilé les principales notions. Les éventuels corrigés de ces exercices sont regroupés dans le chapitre "Corrigés", page 237 en fin de manuel.</p>

4) Commentaires



Des commentaires généraux sur la leçon.
Si vous souhaitez approfondir le sujet, la rubrique “Pour aller plus loin” vous fournit des pistes à étudier.

Vous pourrez également naviguer dans ce guide via les thématiques des chapitres.

1 Premiers pas dans 4D

DÉMARAGE

Thématique

Objectif(s) pédagogique(s)	<ul style="list-style-type: none"> • Comprendre ce qu'est 4D • Découvrir l'interface et l'architecture
Durée estimée	20 minutes
Ce que nous allons utiliser	4D Autoformation Navigation dans les menus Clic sur une base exemple

MISE EN ŒUVRE

Vous devez de télécharger et d'installer votre logiciel et connaître ses bases principales pour...

Cinq thématiques différentes vous sont proposées :

- Utilisation
- Interface
- Programmation
- Plug-ins
- Sécurité

A chaque étape je suis là pour vous guider et vous assister... il faut bien que je travaille un peu aussi :-)

Pour gagner en efficacité, vous pourrez utiliser au début de chaque chapitre, soit la base que vous venez de terminer au chapitre précédent, soit la **base exemple** fournie. Nous pourrons ainsi travailler avec des bases identiques. Les bases exemples sont placées dans le dossier *Bases Autoformation*.

Votre objectif (qui est aussi le mien) est d'aller à l'essentiel et de comprendre l'architecture globale de 4D sans entrer dans tous les détails et possibilités... ce ne serait plus un guide de découverte, mais une encyclopédie !

Vous trouverez à la fin de ce guide les corrigés des exercices ainsi que des sources d'informations utiles pour poursuivre et approfondir votre apprentissage.

PRINCIPES FONDAMENTAUX

Passons immédiatement à la découverte de 4D.

Pour créer votre application professionnelle, vous avez besoin des outils ou concepts suivants (la liste est ouverte...) :

- Modèle de données
- Menus
- Mots de passe
- Images
- Interface (formulaires)
- Méthodes
- Process
- Pointeurs
- Transactions
- ...

On peut les représenter de la manière suivante :



Vous pouvez bien évidemment les implémenter progressivement et tous ne sont pas nécessaires dans toutes vos applications.

D'autre part, votre application doit éventuellement offrir les fonctionnalités suivantes :

- Ajouter, modifier, Supprimer
- Chercher, trier
- Imprimer
- Importer, exporter
- Envoyer des emails
- Accéder au Web (Publier sur le Web)
- Echanger des données (XML, Web Services...) et communiquer (TCP, FTP...)
- S'interconnecter avec d'autres bases de données (ODBC, Oracle, MySQL...)
- ...



En résumé, vous avez besoin de 3 éléments principaux :

- Disposer d'une interface
- Stocker des données
- Programmer un peu

Nous aborderons alternativement ces 3 points au cours des différentes leçons.

PRÉSENTATION DE L'APPLICATION EXEMPLE

Tout au long de ce manuel, nous nous efforcerons de développer une base de données répondant à des besoins spécifiques (ceux que vous rencontrerez très probablement au cours de votre vie de développeur 4D).

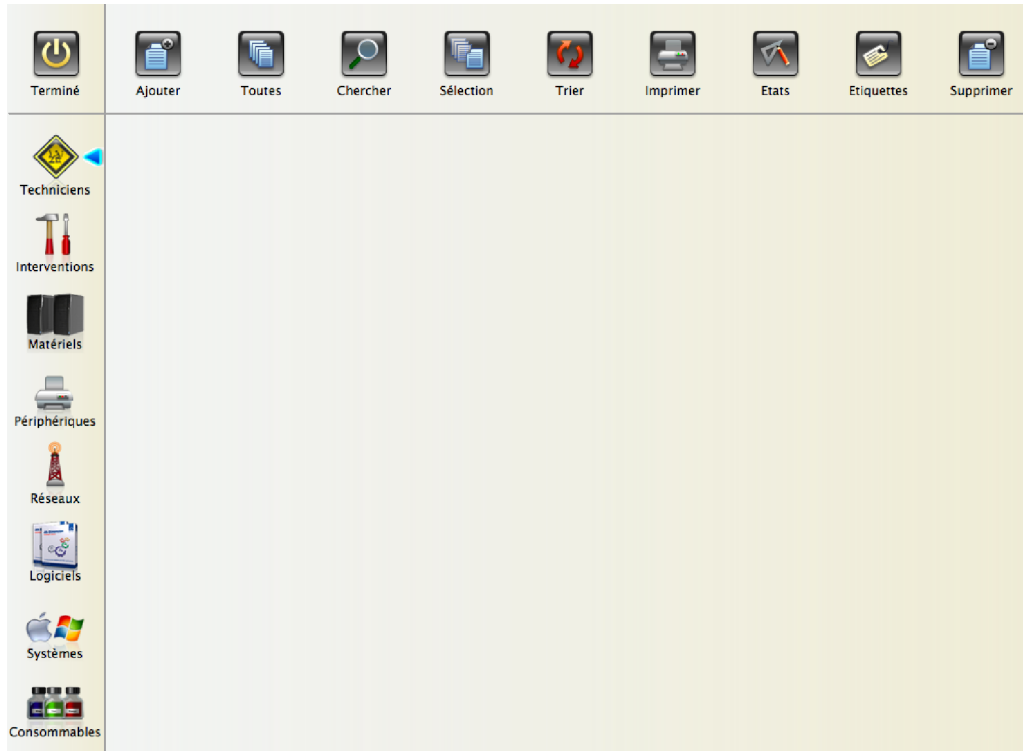
La première étape consiste à identifier ces besoins.

- ▶ Quel est l'objet de l'application qu'on nous demande de développer ?
 - il s'agit une base de données destinée à gérer la maintenance d'un parc informatique
 - elle doit pouvoir gérer les demandes d'interventions sur ce parc
 - ces demandes d'interventions doivent être publiées sur le Web
- ▶ Quelle interface nous a-t-on demandée ?
 - une palette d'outils qui regroupe les fonctions essentielles par thème
 - une barre de menus dans laquelle on trouve un certain nombre de tables et d'options
 - une gestion des préférences par utilisateur

L'étape suivante, constituée des différentes leçons y apportera des solutions.

Rappel : Ce guide n'a pas pour objectif d'aborder l'ensemble des possibilités de 4D, ni d'être un manuel d'analyse (Merise, UML...). Nous essaierons de rester pragmatiques et efficaces.

Dernière information transmise par le client : cette maquette réalisée par son graphiste, suite à quelques réunions... Nous disposons des icônes qu'il a réalisées selon nos préconisations (nous en verrons le détail plus loin).



1 Premiers pas dans 4D

Objectif(s) pédagogique(s)	<ul style="list-style-type: none"> • Comprendre ce qu'est 4D • Découvrir l'interface et l'architecture
Durée estimée	10 minutes
Ce que nous allons utiliser	4D monoposte Navigation dans les menus Essai sur une base exemple

MISE EN OEUVRE

Vous venez de télécharger et d'installer votre logiciel et souhaitez en tirer rapidement parti.

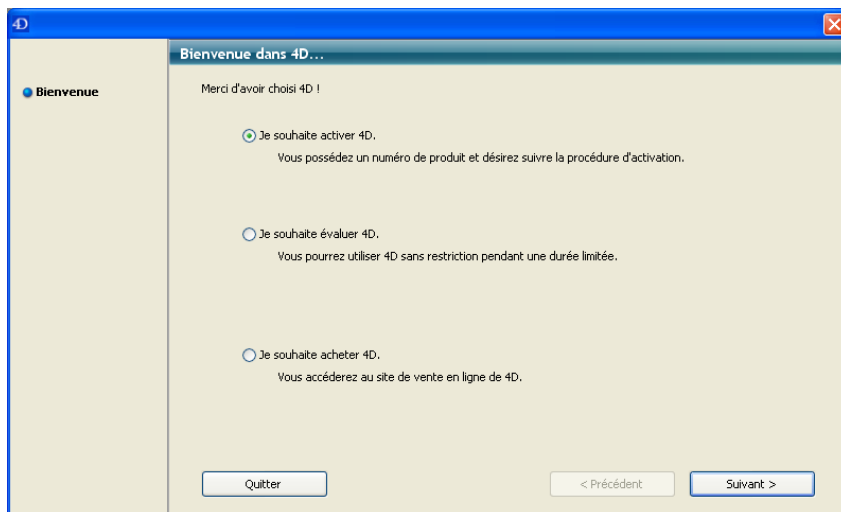
Découvrons l'interface

1 Démarrez le programme 4D.



Si vous n'avez pas encore enregistré votre produit (ou ne disposez pas de votre numéro de produit), vous pouvez cliquer sur le bouton "Je

souhaite évaluer 4D", vous entrerez la licence ultérieurement (voir la procédure dans le *Guide d'installation* de 4D).



4D dispose de 2 modes de travail :

- le mode Développement qui vous permet de concevoir et tester votre développement,
- le mode Application auquel vos utilisateurs ou clients auront accès.

Nous n'utiliserons dans un premier temps que le mode Développement puis découvrirons rapidement la manière de créer un environnement "utilisateurs".

Les quatre premiers menus à gauche de la barre de menus (**Fichier, Edition, Exécution, Développement**) permettent de gérer toutes les fonctions de développement.

Les 2 menus à droite (**Enregistrements** et **Outils**) permettent de tester rapidement les fonctions implémentées.

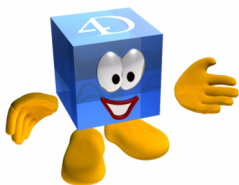
Des menus supplémentaires apparaissent à droite de ces menus en fonction du contexte affiché (formulaire, méthode, outils...)

Lors des manipulations, vous aurez souvent à utiliser la barre d'outils de 4D. Comme dans les menus, les boutons à gauche de la case de recherche servent au développement et les boutons à droite à l'utilisation (manipulation des données)



Vous trouverez dans la documentation de 4D la description détaillée des fonctions associées à ces boutons.

Comme je vous l'ai déjà indiqué, l'exemple que nous suivrons tout au long de ce guide concerne la mise en place d'un logiciel de gestion de la maintenance d'un parc machines. Ouf ! vous avez échappé à la gestion de factures :-)))



J'ai choisi cet exemple car il intègre plusieurs notions (planning, stock, suivi, relances...) transposables à tout type de bases de données professionnelles (gestion de stock, de parc immobilier, CRM, suivi d'interventions, gestion d'ateliers de maintenance, gestion d'équipes, Gestion de la qualité, contrôles réglementaires...) ou personnelles (Vidéothèque, cave à vin, agenda...)

Pour que ce guide reste digeste et agréable, j'ai limité le nombre de fonctionnalités, car l'objectif est bien que vous compreniez les principes sans entrer dans un développement complet.

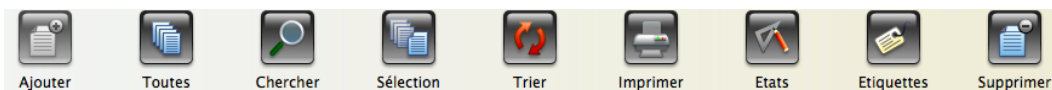
Mais avant de savoir comment faire, regardons ce que ça donne une fois le travail terminé... Vous apprécierez d'autant plus la facilité de mise de oeuvre de bien des fonctionnalités.

2 Dans la barre d'outils, cliquez sur le menu *Ouvrir* et choisissez "GestionParcInfo_36_corrigée.4dbase" dans le dossier "Bases Autoformation".

Dès l'ouverture la fenêtre principale s'affiche et vous présente le contenu de la table Techniciens. Chacune des tables peut contenir jusqu'à quelques milliards d'enregistrements... vous êtes prêts à saisir ?

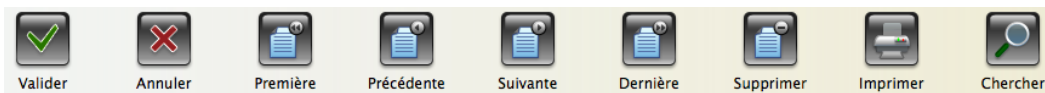
Nous allons découvrir les fonctionnalités prévues dans cette base, en commençant par l'écran principal de navigation.

La barre de boutons en haut permet de manipuler les enregistrements de chacune des tables (ajouter, chercher, trier, imprimer, supprimer...).



L'affichage de chacune de ces tables s'effectue en liste. Il suffit de double-cliquer sur une ligne pour afficher dans la même fenêtre l'écran de détail (contenu détaillé d'une seule fiche).

Lorsque vous double-cliquez sur une ligne, le fiche de DETAIL apparaît avec les boutons de navigation suivants :



Pour afficher les enregistrements d'une autre table, cliquez sur le bouton correspondant dans la barre d'outils sur la partie gauche de la fenêtre.



Le curseur  indique la table sur laquelle on travaille.

Les 2 derniers choix permettent de gérer :

- les préférences (langue...)



- les paramètres (listes de valeurs prédéfinies)



En tête des listes, chaque titre est cliquable et permet de chercher ou trier les données de la colonne.



Vous pourrez rouvrir cette base à tout moment pour retrouver un fonctionnement attendu ou demandé dans les exercices.

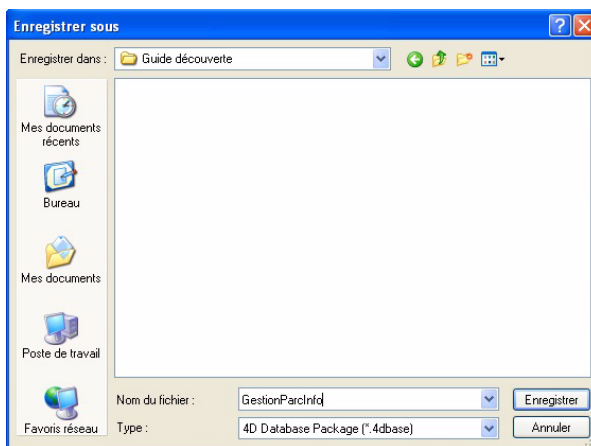
EXERCICE



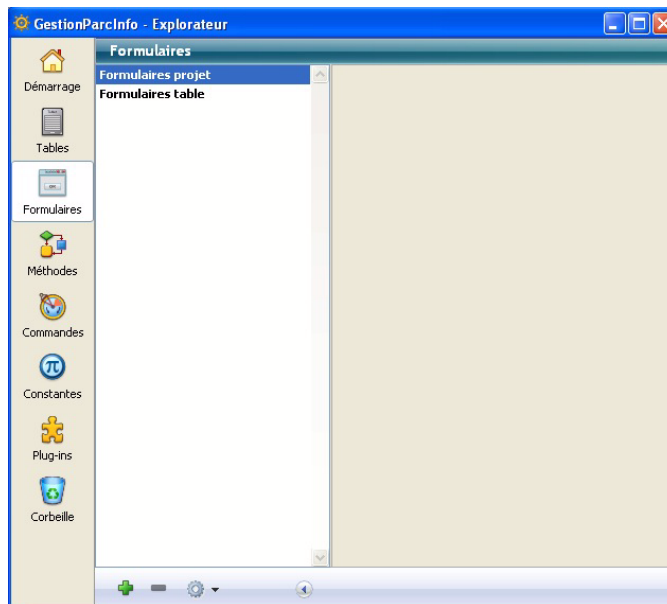
Naviguez quelques instants au sein de la base *GestionParInfo_36_corrigée* pour en découvrir les principales fonctionnalités (bulles d'aides, facilités de saisie, paramétrages, préférences, accès Web...)

Vous allez maintenant créer une nouvelle base de données :

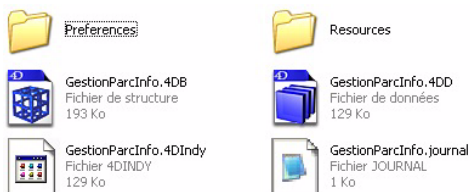
- 1 Dans le menu *Fichier*, choisissez *Nouveau* puis *Base de données*.
- 2 Donnez un nom à votre base (par exemple "GestionParInfo").



Une fois votre base de données enregistrée, 4D vous présente l'Explorateur.



Sur votre disque dur, le dossier *GestionParcInfo* comporte les éléments suivants :



COMMENTAIRES



Cette base correspond à ce que vous obtiendrez à la fin de ce guide. Vous pourrez bien évidemment lui ajouter de nombreuses fonctionnalités (ergonomie, programmation...) pour en faire une application professionnelle.

Je vous conseille de consulter les autres bases exemples ainsi que le site www.4D.fr pour découvrir les nombreuses possibilités et solutions imaginées et mise en œuvre par les ingénieurs développeurs que nous sommes tous devenus :=)))

Et pour ceux qui auront besoin de fonctions ou commandes spécifiques non disponibles en standard dans 4D ou dans les plug-ins, vous pourrez utiliser le Plugin SDK (gratuit) qui vous permettra de réaliser en un clin d'œil la structure de vos projets C++.

Pour aller plus loin

- Paramétrer les sauvegardes automatiques
- Les objets d'interface

2

Formulaires et boutons

Objectif(s) pédagogique(s)	Mise en place et test des premiers éléments d'interface (boutons) sur un formulaire
Durée estimée	10 minutes
Ce que nous allons utiliser	Formulaires, Boutons, Propriétés d'objets

MISE EN OEUVRE

Pour vous familiariser rapidement, commençons par réaliser le maquetage de la navigation.

Procédons par étapes ; nous devons disposer :

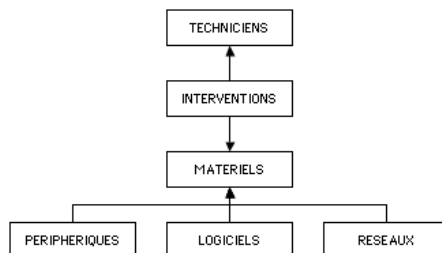
- d'un formulaire d'accueil avec des objets fonctionnels et "décoratifs"
- de quelques lignes de code pour afficher le formulaire à l'écran
- d'écrans de saisie/modification pour chacun des fichiers à traiter

Commençons cette découverte par la réalisation de l'écran principal de navigation. Comme vous l'avez vu, nous devons intégrer les fonctionnalités suivantes dans notre logiciel 4D_Parc :

- Gestion multi sites des utilisateurs et des postes
- Gestion des paramètres personnels
- Gestion des configurations matérielles par poste





- Gestion des configurations réseau par poste
- Gestion des configurations logicielles par poste avec suivi des licences et des versions
- Gestion et suivi des demandes d'intervention
- Gestion des périphériques (imprimantes, scanners, appareils photo, vidéo projecteurs, etc.)

Les interdépendances sont matérialisées par le schéma ci-dessous :



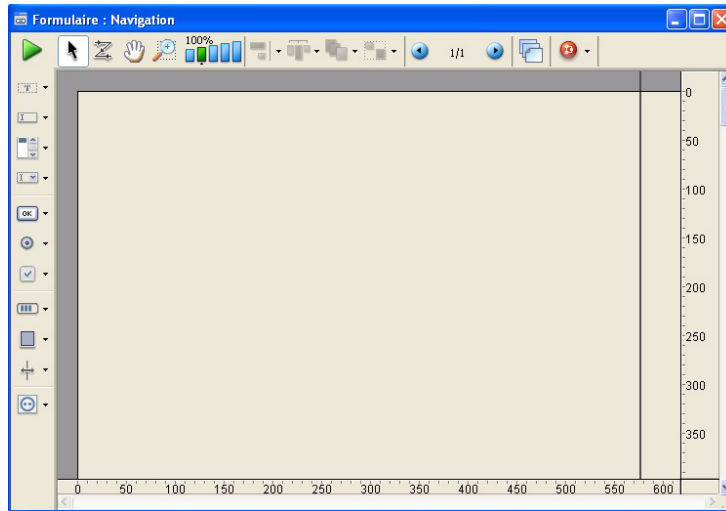
Vous voyez, nous allons rester dans une structure simple et suffisante pour découvrir 4D.

Dans la barre d'outils :

- 1 Cliquez sur le bouton "Explorateur"  .
- 2 Cliquez sur le bouton "formulaires" Windows  Mac OS  .
- 3 Choisissez "Formulaires projet" puis cliquez sur le bouton  en bas de l'explorateur.
- 4 Saisissez le nom "Navigation".

- 5 Cliquez sur OK.

Un formulaire vide apparaît :




Les outils à gauche permettent d'ajouter des objets sur le formulaire (boutons, variables, champs, tableaux, textes, plug-ins...).

Les outils en haut permettent de manipuler les objets (grouper, aligner, répartir...) et de gérer les options d'affichage au sein du formulaire (zoom, pages, vues...)

Le triangle ▼ à droite des outils indique que plusieurs options sont accessibles pour l'outil

Commençons par créer une interface de navigation simple pour en comprendre le fonctionnement :

- un bouton pour afficher la liste des Techniciens,
- un bouton pour fermer le formulaire,
- nous ajouterons un titre en haut de la page.

1 Sélectionnez l'objet bouton .

2 Tracez un premier bouton sur le formulaire.

La liste des propriétés apparaît.

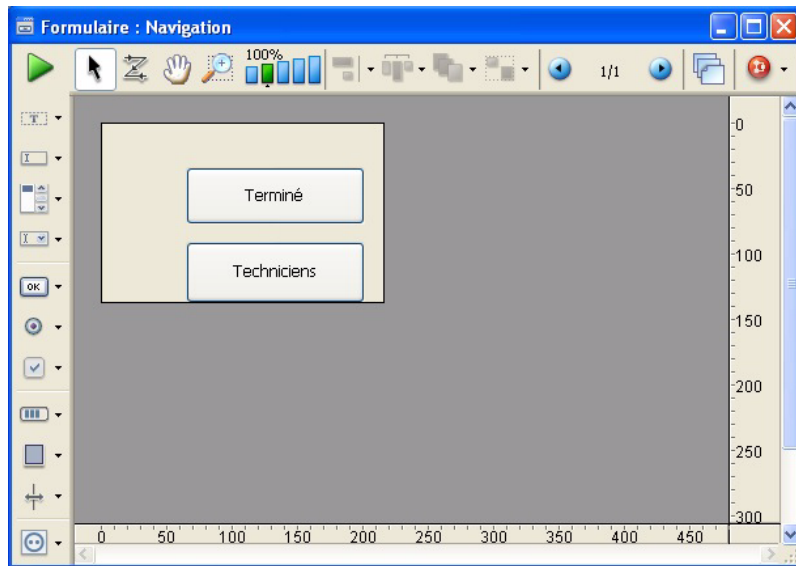


ASTUCE

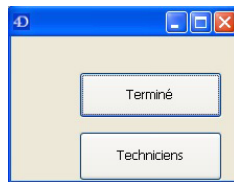
Pour faire réapparaître la liste des propriétés lorsque vous l'aurez fermée, double-cliquez sur le bouton ou effectuez clic droit + liste des propriétés.

3 Renommez le bouton "Terminé" en modifiant la propriété Titre dans le thème "Objets".

- 4 Appuyez sur Entrée pour valider votre modification.
- 5 Cliquez sur "pas d'action" dans la propriété "Action Standard" dans la partie inférieure de la liste des propriétés puis choisissez "Valider".
Ce choix permettra de refermer la fenêtre.
- 6 Créez un 2e bouton et renommez-le "Techniciens".
Nous verrons bientôt comment programmer ce bouton pour qu'il affiche la liste des techniciens.
- 7 Placez les boutons comme ci-dessous :



Pour tester votre oeuvre, cliquez sur le triangle vert en haut à gauche de la fenêtre du formulaire. 4D vous présente le formulaire tel que l'utilisateur le verra.



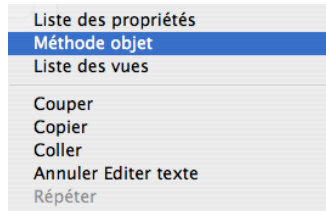
Vous remarquez que la fenêtre s'est adaptée toute seule à la taille du formulaire. Le redimensionnement sera automatique et vous fera gagner de temps en développement et en maintenance (cet automatisme de redimensionnement est bien évidemment débrayable).

8 Cliquez sur le bouton "Terminé" pour refermer la fenêtre et revenir en mode Développement.

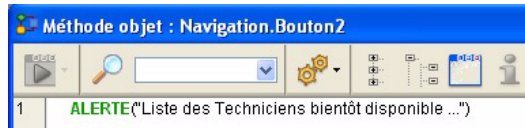
Vérifions que le 2e bouton est fonctionnel :

1 Effectuez un clic droit sur le bouton "Techniciens".


2 Choisissez "Méthode objet" :



3 Dans la fenêtre de la méthode objet, saisissez la ligne suivante : ALERTE("Liste des Techniciens bientôt disponible...")

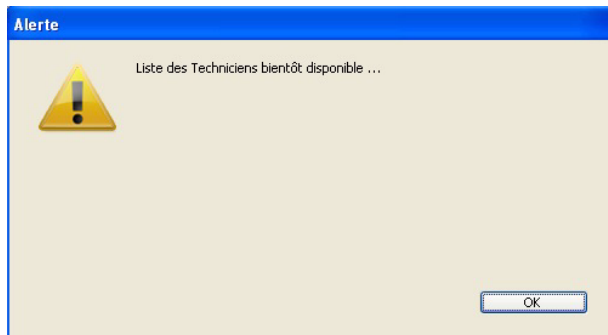


4 Refermez la fenêtre de méthode.

Vous remarquerez le "badge"  qui indique qu'une méthode est associée à l'objet.

5 Retestez votre formulaire puis cliquez sur le bouton "Techniciens".

Le message suivant doit apparaître :



Comme vous pouvez le constater, c'est très simple à mettre en œuvre... Vous voici donc autonome et prêt pour les exercices :-)

Dans une leçon ultérieure, nous transformerons ces boutons en boutons images afin d'améliorer l'ergonomie de notre application.

EXERCICE



Pour confirmer cette première approche, créez un deuxième bouton destiné à afficher la liste des "Interventions".

Puis terminez le formulaire pour qu'il ressemble à l'exemple proposé.



Voir corrigé page 237.

COMMENTAIRES



Lorsque vous dupliquez un bouton (copier-coller), vous dupliquez également l'ensemble de ses propriétés ainsi que sa méthode associée. (NB : Seul le nom d'objet est différent car unique par formulaire).

Je vous conseille donc en général de peaufiner un objet initial et de le dupliquer ensuite pour ne changer que les quelques paramètres qui le concernent.

Pour aller plus loin

- Détail des objets de formulaire
- Gérer des vues

3 Affichage au démarrage

Objectif(s) pédagogique(s)	Affichage d'un formulaire dès l'ouverture de la base. Création d'une méthode projet appelée depuis la méthode base "Sur ouverture"
Durée estimée	5 minutes
Ce que nous allons utiliser	Méthode base, programmation simple, opérateurs

MISE EN OEUVRE

Avant de continuer, découvrons les règles de syntaxe de base, ainsi que les principaux opérateurs utilisés dans 4D :

Traitement ou manipulation		Opérateur	Exemple
Comparaisons	Egal	=	Si(Valeur = 12)
	Différent de	#	Si(vtexte # "4D")
	Supérieur Inférieur	> <	Si(Valeur1 > Valeur2)
	Supérieur ou égal Inférieur ou égal	>= <=	Si(Valeur1 >= Valeur2)

Affichage au démarrage

Conjonctions de coordination (dans le cadre des recherches, des tests...)	ET	&	Si((\$Euros=Riche) & (\$Esthet=Beau))
	OU	 Ce caractère est obtenu par : • Alt-GR6 sur Windows • Alt-Maj-L sur Mac OS	Si((il_pleut) (il_neige))
	SAUF	# utilisé seulement dans le cadre des recherches	
Valoriser un champ ou une variable	Affectation	:=	vNom:="Napoléon"
Opérations sur les numériques	Addition Soustraction Multiplication Division Exponentiation Parenthèses	+ - * / ^ ()	vTotal:=vHT+vTTC vRevenus:=vSalaires-vRetenues-vImpôts vTotal:=vNbHeures*vTauxHoraire vLongueur:=vSurface/vLargeur vKiloOctet:=2^10 vTotal:=(5+3*(2-7)-((4*5)/(2+4)))
Opérateurs sur les chaînes	Concaténation Multiplication Indice de chaîne	+ * [[]] (Windows) et ≤ ≥ (Mac OS)	vNom:="Napoléon"+" "+"Bonaparte" vTest:="x"*4 (on obtient "xxxx") vNom[[1]]:=Majusc(vNom[[1]])
Opérateurs sur les dates	Addition	+ ou -	VNbJours:=!06/05/2007! - !22/04/2007!





Pour obtenir des renseignements complémentaires sur les opérateurs sur les images, les chaînes, etc. reportez-vous à la documentation de 4D (thème *Opérateurs*).

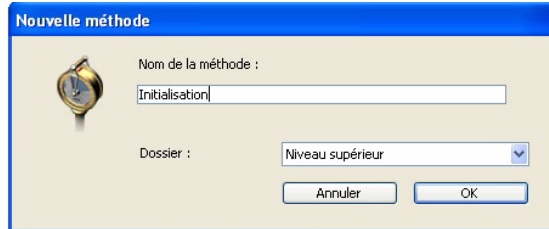
Ces bases sont maintenant posées... et connues :-)), nous pouvons continuer !

Notre application devra afficher l'écran de navigation dès que l'utilisateur démarrera le programme. Il faut donc rendre automatique l'affichage de ce formulaire au démarrage de la base.

Créons une première méthode en utilisant l'Explorateur, ce sera une "Méthode Projet".

1 Affichez l'Explorateur .

- 2 Choisissez "Méthodes" (4^e icône à gauche de l'explorateur) Windows  Mac OS  .
- 3 Cliquez sur le bouton Windows  MacOS  en bas de l'explorateur.
- 4 Nommez la méthode "Initialisation" puis cliquez sur OK.



- 5 Recopiez le code suivant :



```


1 $NumFenetre:=Creer fenetre formulaire("Navigation")
2 DIALOGUE("Navigation")
3 FERMER FENETRE($NumFenetre)

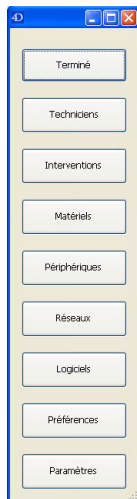
```

La première ligne crée un conteneur dimensionné à la taille du formulaire Navigation, en tenant compte des propriétés que nous avons définies.


La 2e ligne affiche le formulaire "Navigation" dans le conteneur.

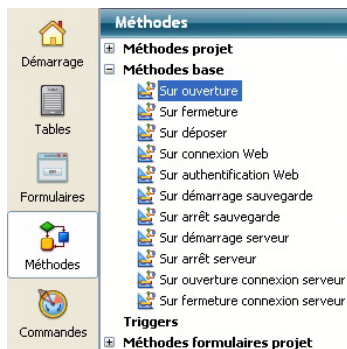
La 3e ligne est exécutée lorsqu'on ferme la fenêtre en cliquant sur le bouton **Terminé**. Elle referme le conteneur.

Pour vérifier le bon fonctionnement de cette méthode, cliquez sur le bouton Exécuter  en haut à gauche de la fenêtre de méthode. Votre formulaire doit apparaître dans une fenêtre "taillée sur mesure" :



L'affichage automatique au démarrage est presque terminé. Il faut maintenant "appeler" cette méthode lors du démarrage de 4D :

- 1 Affichez l'explorateur  .
- 2 Ouvrez la liste des "méthodes base".
- 3 Double-cliquez sur la méthode base "Sur ouverture".



Cette méthode s'exécute automatiquement au démarrage. Pour ne pas ressaisir le code déjà écrit, nous allons "appeler" la méthode "Initialisation" :

- 1 Ecrivez simplement les premières lettres du nom de la méthode à appeler (Init) puis appuyez sur la touche tabulation.

4D reconnaît que c'est une méthode et l'affiche en italique bleu et gras).

2 Refermez la méthode base "Sur ouverture"

Pour tester :

1 Choisissez le menu "Fichier" puis "Ouvrir bases récentes".

2 Sélectionnez le nom de votre base de données.

La base se referme puis se rouvre. Vous devez obtenir d'emblée le formulaire dans une fenêtre en mode utilisateur.

Cliquez sur "Terminé", le mode Développement réapparaît et affiche les fenêtres qui étaient restées ouvertes en quittant.

Bravo !! Cette étape est terminée, nous allons continuer en améliorant l'interface de notre formulaire... elle en a bien besoin :=)

EXERCICE



Mettez en place un système qui permet d'afficher le message " A bientôt" lorsqu'on quitte la base de données.

Indice : cet exercice requiert l'utilisation de la méthode base Sur fermeture.



Voir corrigé page 237.

COMMENTAIRES



Une méthode projet peut être appelée par toute autre méthode (objet, formulaire, base, projet, trigger, plug-ins, ligne de menus...). Une méthode peut s'appeler elle-même, dans ce cas on parle de méthode récursive.

Dans les méthodes base je vous conseille de ne faire appel qu'à des méthodes projets. Vous conservez ainsi la possibilité d'exécuter ou d'appeler la méthode quand bon vous semble sans avoir à quitter et redémarrer votre base.

Affichage au démarrage

Nous verrons plus loin dans ce guide comment passer des paramètres d'une méthode à une autre.

Pour aller plus loin

- Bien comprendre les méthodes base
- Les différents types de méthodes

4

Barres de menus

Objectif(s) pédagogique(s)	Mettre en place une interface utilisant des menus
Durée estimée	5 minutes
Ce que nous allons utiliser	Barre de menus, programmation simple


Bien que l'interface soit actuellement plutôt orienté "presse-boutons", les barres de menus restent un outil efficace auquel vos utilisateurs sont habitués.

Dans 4D, vous pouvez créer jusqu'à 32000 barres de menus. Chaque barre peut être nommée afin de s'y retrouver plus facilement.

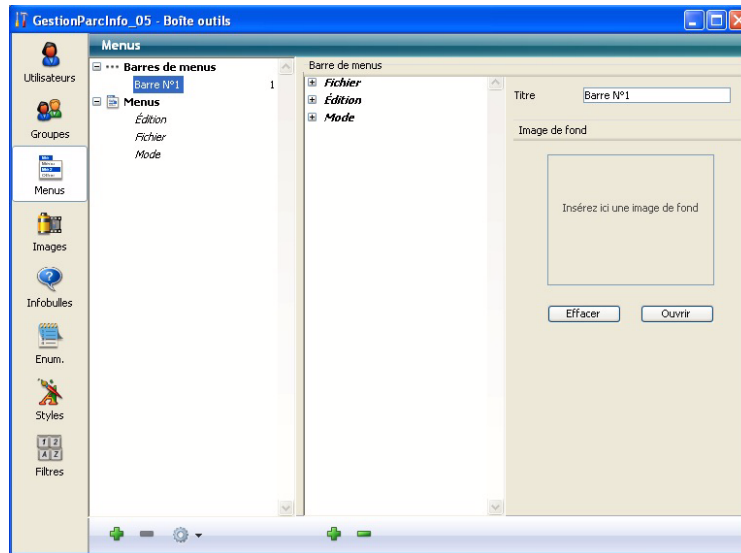
Une barre de menu peut être affichée par programmation (commande FIXER BARRE MENUS) ou parce qu'un formulaire lui est associé. Dans ce cas, lorsque le formulaire apparaît, la barre de menu remplace la précédente.

MISE EN OEUVRE

Maintenant que l'ensemble fonctionne, nous allons créer une barre de menus et appeler cette méthode à partir d'une ligne de menus.

- 1 Affichez la boîte à outils (bouton  dans la palette).

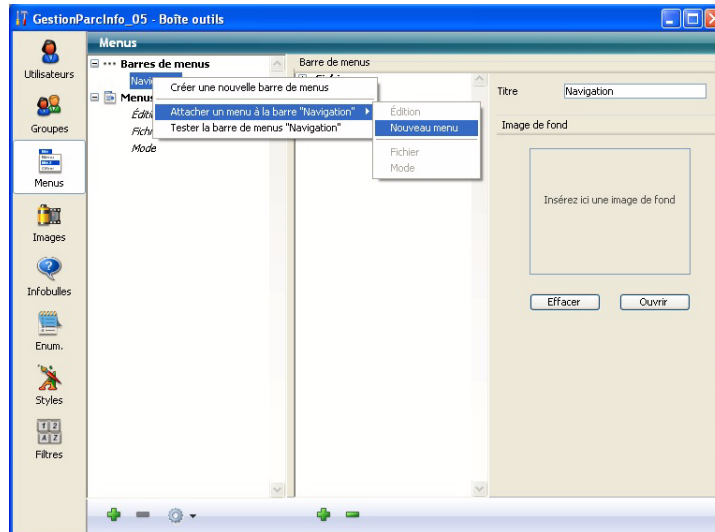
2 Sélectionnez "Menus".



Cette fenêtre présente en premier la liste de toutes les barres de menus et lorsque vous cliquez sur une barre, l'ensemble de ses menus. Ensuite par menu vous pouvez afficher chaque ligne. La partie droite de l'écran présente le détail des éléments sélectionnés.

3 Renommez la Barre N°1 "Navigation" (en haut à droite de la fenêtre dans Titre ou en double-cliquant sur son nom dans la liste).

- 4 Affichez le menu contextuel (clic droit sur la barre Navigation), choisissez *Attacher un menu à la barre Navigation* puis *Nouveau menu* :



- 5 Renommez ce menu "Outils".
- 6 Affichez le menu contextuel de ce nouveau menu et choisissez *Ajouter une ligne au menu Outils*.
- 7 Renommez cette ligne "Tableau de bord".
- 8 Choisissez la méthode "INITIALISATIONS" dans la liste déroulante sur la partie droite de l'écran. Le nom de la méthode sélectionnée vient s'inscrire à droite de la ligne.

Pour associer ce nouveau menu à la barre Navigation, vous pouvez également procéder de cette manière :

- 1 Cliquez sur la barre Navigation
- 2 Dans la colonne de droite qui contient les menus actuels de la barre, affichez le menu contextuel (clic droit)
- 3 Choisissez "Attacher un menu à la barre Navigation".
- 4 Cliquez sur le menu Outils, il est attaché automatiquement.

Pour tester cette nouvelle possibilité d'accès, choisissez **Tester l'application** dans le menu **Exécuter**. Vous voyez apparaître votre menu et pouvez tester l'efficacité de la ligne que vous avez créée.

EXERCICE



Dans la suite des leçons, vous pourrez ajouter une ligne de menu dans ce menu **Outil** pour chaque fonctionnalité particulière.

COMMENTAIRES



Vous pouvez activer/désactiver les lignes de menus en fonction des manipulations réalisées par l'utilisateur.

Vous aurez également remarqué que les menus standard font appel à une ressource :xliff:nom. Cette notation indique que le titre du menu provient du fichier de paramétrage nommé *CommonFR.xlf* situé dans le dossier */Contents/Resources/French.lproj*.

Ce principe vous permet de comprendre l'interaction qui peut exister entre un fichier de ressources (en XML dans le cas présent) et votre base de données. Une application évidente concerne la mise en oeuvre d'applications multilingues.

Attention, ne modifiez pas la ressource indiquée ci-dessus car elle est gérée et utilisée par 4D. Vous risqueriez de générer des dysfonctionnements. Je ne l'ai évoquée qu'à titre d'exemple.

Pour aller plus loin

Dans le thème "Menus", vous trouverez de nombreuses commandes qui permettent par exemple de :

- modifier la méthode associée à une ligne de menu
- modifier le raccourci d'une ligne
- connaître par programmation (Menu choisi) le menu et la ligne choisis par l'utilisateur

5

Page 0 et page 1

Objectif(s) pédagogique(s)	Structurer un formulaire pour qu'il affiche certains objets sur toutes les pages
Durée estimée	20 minutes
Ce que nous allons utiliser	Page 0 et 1, bibliothèque d'images

MISE EN OEUVRE

Les formulaires de 4D sont multipages (32767 maximum), ce qui nous permet de répartir les informations de manière thématique ou fonctionnelle, puis d'accéder à telle ou telle page en fonction de nos besoins. La navigation entre les pages peut se faire en utilisant des onglets, des boutons...

Certains objets (textes, boutons...) devront cependant être visibles quelle que soit la page affichée.

Dans ce cas, il faut mettre ces objets sur la page 0 que 4D affiche systématiquement "en dessous" de la page courante (1, 2...).

Par exemple tous les boutons à gauche et en haut dans la base exemple finalisée figurent sur la page 0.

Pour naviguer de page en page ou en ajouter, vous pouvez utiliser les boutons gauche et droite dans les outils en haut de la fenêtre du

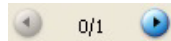
formulaire ou utiliser le pop-up menu entre les deux flèches qui permet d'accéder à une des pages existantes.



Pour déplacer des objets d'une page à une autre, il suffit de les couper, puis de les coller sur la page souhaitée.

1 Déplacez l'ensemble des boutons de votre formulaire sur la page 0.

L'indicateur de page devient :



Nous allons maintenant remplacer tous ces boutons au look "quelconque", enfin... disons... "système", quoique pleinement fonctionnels, par de superbes boutons ergonomiques et esthétiques dignes d'une interface moderne...

2 Double-cliquez sur le bouton "Terminé" pour afficher ses propriétés.

Profitons-en pour lui donner un nom de variable plus parlant que "Bouton1" ... par exemple "B_Fin".

Type	Bouton
Nom de l'objet	Bouton1
Nom de la variable	b_Fin
Titre	Terminé

3 Cliquez sur le mot "Bouton" dans la propriété "Type" (1ère de la liste) et choisissez "Bouton Image".

Type	Bouton image
Nom de l'objet	Bouton
Nom de la variable	Bouton 3D
Image	Bouton invisible
	Bouton inversé
Source	Bouton image
Nom/N°	Grille de boutons
Effet miroir (Windows)	Case à cocher
Découpage	Case à cocher 3D
	Bouton radio
Lignes	Bouton radio 3D
Colonnes	Bouton radio image

Le bouton s'est transformé en :





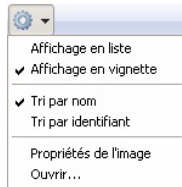
Si vous observez les autres propriétés, vous voyez, juste en-dessous de son nom, les propriétés relatives à l'image (provenance et numéro de référence) :

Image	
Source	Fichier de ressources
Nom/N°	806

A ce stade, aucune image ne figure dans la bibliothèque d'images. Nous ne pouvons pas indiquer de numéro ou de nom d'image.

Voyons maintenant comment ajouter une image dans la bibliothèque.

- 1 Ouvrez la boîte à outils .
- 2 Cliquez sur "Images" .
- 3 Cliquez sur le menu déroulant en bas et choisissez *Ouvrir* :



- 4 Choisissez l'image "Fermer.png" dans le dossier "Images_PNG" du dossier "Bases_Autoformation".



Vous pouvez également glisser les images directement du dossier vers la liste de l'éditeur d'image, 4D vous proposera de définir les caractéristiques de l'image glissée.

Vous remarquez qu'il est réalisé avec 4 images superposées qui représentent les 4 états d'un bouton (affiché, cliqué, survolé, désactivé) :



La fenêtre des propriétés de l'image s'affiche :

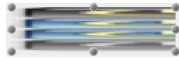
- 5 Donnez un nom à l'image (son numéro et ses dimensions sont automatiques mais modifiables à ce stade). Indiquez également que l'image est composée de 4 images (1 colonne sur 4 lignes) :

- 6 Mémorisez le numéro d'image et la taille (48x48) puis cliquez sur OK.

Votre première image est importée. Vous pouvez maintenant refermer la Boîte à outils et revenir à votre formulaire pour indiquer la source et le numéro d'image que le bouton doit utiliser.

Image	
Source	Bibliothèque d'images
Nom/N°	1

Votre bouton a bien changé ! Rassurez-vous, nous allons l'arranger un peu.



7 Changez la taille et les propriétés du bouton de la manière suivante :

- 4 lignes sur 1 colonne :

Découpage	
Lignes	4
Colonnes	1

- Bascule sur passage du curseur, Retour sur relâchement du clic et Dernière imagerie si désactivé :

Animation	
Défilement continu sur clic	<input type="checkbox"/>
Recommencer la séquence	<input type="checkbox"/>
Bascule sur passage du c...	<input checked="" type="checkbox"/>
Retour sur relâchement ...	<input checked="" type="checkbox"/>
Dernière imagerie si dés...	<input checked="" type="checkbox"/>
Défilement tous les n ticks	<Aucun>

- Largeur et hauteur à 48 :

Coordonnées et dimensions	
Gauche	15
Haut	27
Droite	63
Bas	75
Largeur	48
Hauteur	48

Notre bouton vient de finir sa cure de jouvence :



8 Testez votre formulaire (▶).

Vous constatez qu'en survolant le bouton ou en cliquant dessus, l'image change.

Vous voici maintenant armé pour implémenter une interface esthétique.



ASTUCE

Pour gagner du temps lorsque vos images figurent déjà dans votre bibliothèque d'images, vous pouvez directement glisser le nom de l'image de la bibliothèque vers votre formulaire. 4D crée automatiquement un bouton image à la bonne taille. Il vous faut juste paramétrer son comportement.

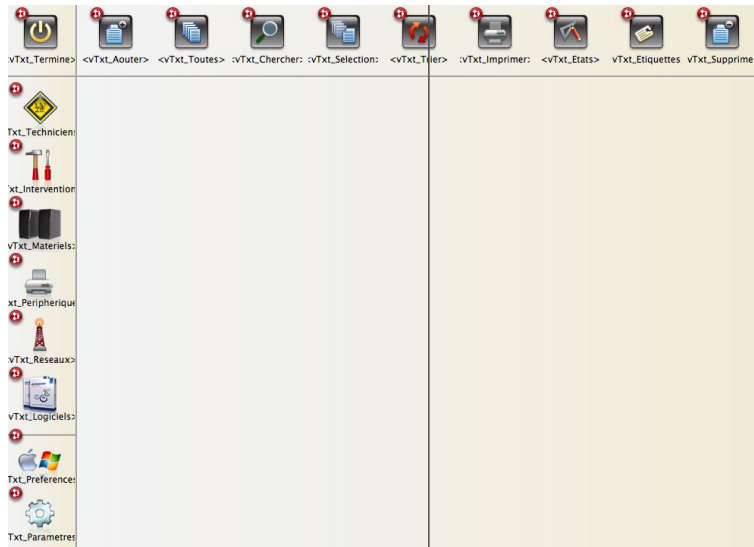


CONSEIL D'une manière générale, tous les objets (méthodes, tables, champs, formulaires, images, etc.) qui figurent dans votre base 4D peuvent être récupérés dans toute nouvelle base par simple transfert. Vous n'aurez ainsi pas à recréer vos images, boutons et autres éléments à chaque nouvelle base.

EXERCICE



- Répétez ces manipulations avec chacun des boutons du formulaire.
- Ajoutez également les boutons fonctionnels (recherche, tri...) comme sur l'exemple proposé.



Il regroupe les différents accès aux tables à gauche et les boutons de fonction (ajouter, imprimer, supprimer...) en haut.

Ne mettez pas de programmation dans ces derniers boutons, nous le ferons lors de leur utilisation.

Remarque : Tous les boutons de la colonne de gauche (de technicien à paramètres) restent en 1 ligne x 1 colonne. Les cases à cocher dans les propriétés d'animation sont :

Animation	
Défilement continu sur clic	<input type="checkbox"/>
Recommencer la séquence	<input checked="" type="checkbox"/>
Bascule sur passage du c...	<input checked="" type="checkbox"/>
Retour sur relâchement ...	<input checked="" type="checkbox"/>
Dernière imagerie si dés...	<input type="checkbox"/>
Défilement tous les n ticks	<Aucun>

Tous les boutons en haut (de Terminer à Supprimer) sont découpés en 4 lignes x 1 colonne et ont comme propriétés d'animation :

Animation	
Défilement continu sur clic	<input type="checkbox"/>
Recommencer la séquence	<input type="checkbox"/>
Bascule sur passage du c...	<input checked="" type="checkbox"/>
Retour sur relâchement ...	<input checked="" type="checkbox"/>
Dernière imagerie si dés...	<input checked="" type="checkbox"/>
Défilement tous les n ticks	<Aucun>

- Créez les textes de ces boutons. Ils seront paramétrés ultérieurement pour tenir compte de la langue choisie.

COMMENTAIRES



Il est souvent tentant (car plus rapide a priori) de coller une image dans plusieurs formulaires plutôt que de l'inclure dans la bibliothèque d'images. Gardez présent à l'esprit que toute modification de l'image dans la bibliothèque est instantanément répercutée sur l'ensemble de la base et ne consomme qu'une fois la taille de l'objet.

Une duplication d'images est difficilement maintenable si elle figure sur 10, 20 ou 80 formulaires. D'autre part, dans ce cas vous augmentez la taille de votre fichier de structure de N fois la taille de l'image.

Exemple : notre image père 8ko, nous avons une dizaine d'images de taille identique = 80 Ko, si on multiplie par 50 formulaires, on occupe 4 Mo sans raison dans la structure.

Pour aller plus loin

Maintenant que ce concept de pages est bien assimilé, vous pourrez commencer à "penser" programmation générique, en tenant compte de la page sur laquelle travaille l'utilisateur en adaptant par exemple les menus, comme nous l'avons vu au chapitre précédent. Dans les chapitres suivants, nous aborderons une extension de ce concept avec la notion d'héritage de formulaires.

6 Pages de formulaire et navigation

Objectif(s) pédagogique(s)	<ul style="list-style-type: none">• Ajouter des pages de formulaire• Mettre en place un système de navigation au sein du formulaire
Durée estimée	5 minutes
Ce que nous allons utiliser	Pages de formulaire, navigation par programmation

MISE EN OEUVRE

Nous disposons maintenant d'une base esthétiquement correcte.

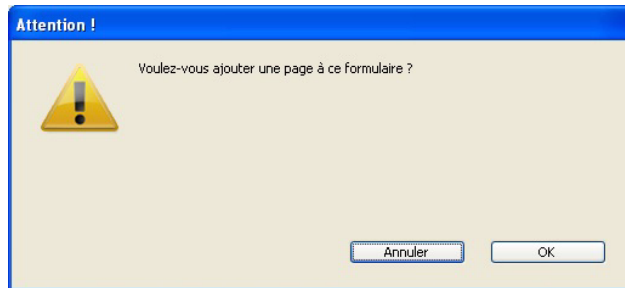
Dans la base exemple que vous venez d'ouvrir, nous avons ajouté en page 0 les boutons de "manipulation" en haut du formulaire (Ajouter, Tout sélectionner, Chercher, Imprimer...).

Nous devons la rendre fonctionnelle en créant les pages 2 à 8 puis en positionnant les informations sur chacune de ces pages et en rendant la navigation possible entre ces différentes pages.

Pour créer les pages :

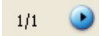
- 1 Affichez le formulaire Projet "Navigation".
- 2 Choisissez "Ajouter page" dans le menu "Formulaire".

3 Confirmez le message d'ajout :



4 Répéter l'opération pour que votre formulaire ait 8 pages (9 en tout si on compte la page 0).



Vous pouvez également cliquer sur la flèche droite du panneau de navigation .

Il faut maintenant indiquer pour chaque bouton la page du formulaire à afficher.

1 Ouvrir la méthode du bouton "Techniciens".

2 Effacez son contenu.

3 Saisissez la ligne suivante :

ALLER A PAGE(1)



4 Reproduisez les étapes 1 à 3 avec le bouton "Interventions", en passant le numéro de page à 2.

5 Ajoutez un texte ou un dessin différent (ou à un emplacement différent) sur chacune des pages.

Cela permettra de contrôler visuellement le changement effectif de page.

6 Testez votre formulaire et sa nouvelle navigation.

Notre formulaire de navigation correspond aux attentes des utilisateurs. Nous devons maintenant leur présenter des données, c'est l'objet du chapitre suivant.

EXERCICE



- Reproduisez la manipulation pour chacun des boutons en modifiant le numéro de page.
- Pour vérifier le changement de pages, ajoutez un texte ou un dessin différent (ou à un emplacement différent) sur chacune des pages.



Voir corrigé page 238.

COMMENTAIRES



Au lieu de gérer dans chaque bouton la même ligne de code, on peut appeler une méthode et lui passer en paramètre le numéro de page à afficher. Ce sera beaucoup plus souple en maintenance et permettra éventuellement de conditionner le changement de page en fonction d'un ou plusieurs critères.

On peut également gérer plus de pages que de boutons visibles et conserver ainsi des informations "masquées" accessibles également "sous conditions".

Pour aller plus loin

- Navigation par onglet
- Gérer les onglets en vue de la compilation
- Naviguer avec la grille de boutons

7

Tables et champs

Objectif(s) pédagogique(s)	Création de tables, de champs et paramétrage des propriétés afin de gérer et conserver les données
Durée estimée	10 minutes
Ce que nous allons utiliser	Editeur de structure, tables, champs

MISE EN OEUVRE

Pour conserver les données d'une session à l'autre ou les partager entre plusieurs utilisateurs, nous pouvons les stocker par exemple dans :

- une table
- un fichier texte
- un fichier XML
- les ressources
- ailleurs (autres bases de données)

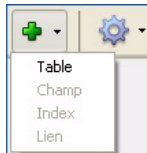
Prenons le cas d'un stockage dans une table. Il nous faut :

1. Créer la table
2. Créer les champs
3. Créer au moins un formulaire de saisie / visualisation

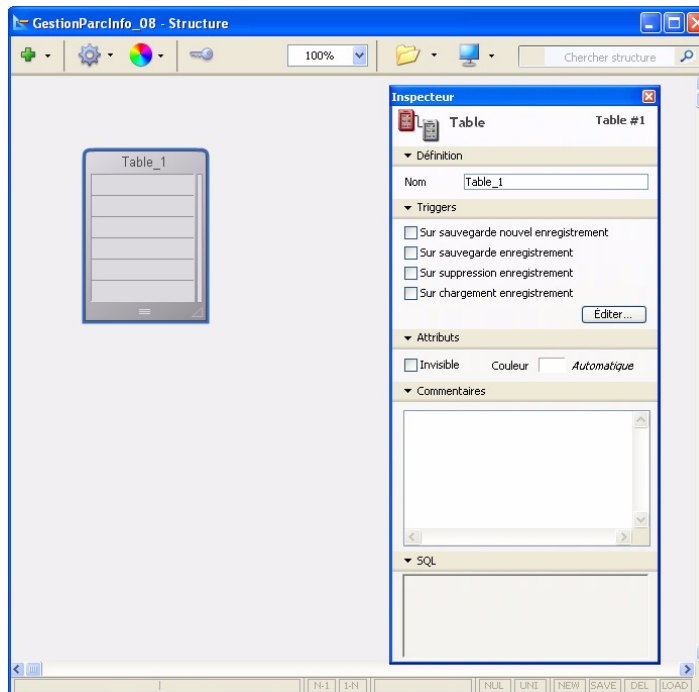
► Pour créer une table :

- 1 Cliquez sur l'icône "Structure"  .

- 2 Cliquez sur le bouton  en haut à gauche de la fenêtre de structure.
- 3 Choisissez "Table".

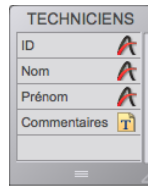


Une nouvelle table est créée dans la structure et affiche l'Inspecteur.

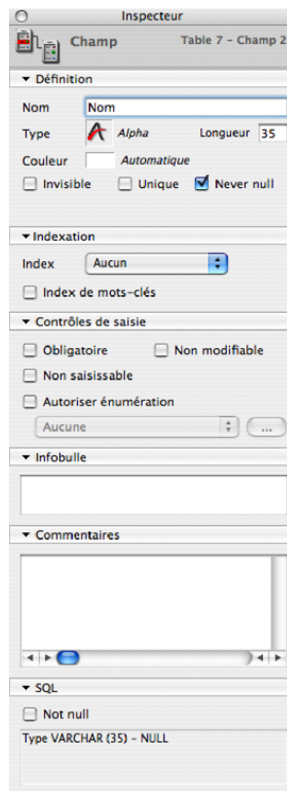


- 4 Nommez la table **TECHNICIENS**.
 - Pour ajouter des champs, plusieurs méthodes existent, prenons la plus rapide :
 - 1 Double-cliquez sur la première ligne de la table.
 - 2 Saisissez le nom du premier champ (ne changez pas le type, nous y reviendrons ultérieurement de manière globale).
 - 3 Validez avec la touche Entrée, un nouveau champ est créé.

4 Continuez ainsi pour tous les champs à créer.

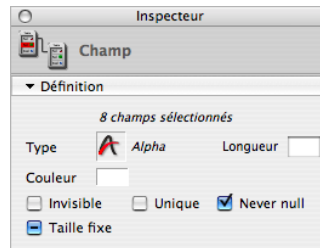


- Vous pouvez revenir sur chacun des champs et sélectionner à droite de son nom le type désiré
- Vous pouvez également double-cliquer sur les champs pour afficher l'Inspecteur et modifier plus finement leurs attributs.



L'infobulle sera affichée lorsque l'utilisateur survolera le champ. Les commentaires permettent de préciser toute information que vous jugez utile (éléments de réflexion, utilité du champ...)

Si vous sélectionnez plusieurs champs, l'Inspecteur présente les éléments relatifs à l'ensemble des champs et précise le nombre de champs concernés :



Dans le chapitre suivant, nous allons créer les formulaires et commencer la saisie.

EXERCICE



Créer de la même manière les tables "Interventions" et "Matériel" selon le modèle ci-dessous :

INTERVENTIONS	
ID	
ID_Technicien	
Objet	
Descriptif	
Date_Debut	
Heure_Debut	
Date_Fin	
Heure_Fin	
Commentaire	
Avancement	
Type_Intervention	
ID_Matériel	

MATERIELS	
Référence	
Description	
Modèle	
Utilisateur	
Département	
Prix Achat	



Voir corrigé page 238.

COMMENTAIRES




Pour modifier le type de plusieurs champs :

- Sélectionnez plusieurs champs contigus (Majuscule) ou non-contigus (touche Ctrl Windows ou Cmd sur MacOS)
- Changez le type d'un des champs => tous les champs sélectionnés sont modifiés

Pour déplacer un champ :

- Appuyez sur la touche Alt (le curseur se transforme en main)
- Glissez le champ à sa nouvelle position

Plusieurs solutions existent pour ajouter des tables et des champs :

- Clic droit (dans la structure pour ajouter une table, dans la table pour ajouter un champ)
- Menu Fichier -> Nouveau
- Bouton "Nouveau" de la barre d'outils
- Bouton  de la fenêtre de structure
- Import de données
- Récupération d'une base existante



CONSEIL

Vous gagnerez beaucoup de temps en créant vos tables à partir de fichiers texte (txt, csv, dbf...) déjà existants. Je vous conseille de regarder la documentation de 4D sur ce sujet (import de données).

Pour aller plus loin

Pour apprendre à concevoir vos bases de données et mettre en place une démarche qualité dans vos projets, participez aux formations suivantes :

- Comment concevoir votre base de données
- Gérer la qualité des projets

8

Saisie et modification d'enregistrements

Objectif(s) pédagogique(s)	<ul style="list-style-type: none"> • Créer rapidement les formulaires de saisie à l'aide de l'éditeur de formulaires • Commencer à entrer des données.
Durée estimée	10 minutes
Ce que nous allons utiliser	Mode "manipulation de données", ajout d'enregistrements

Nous intégrerons par la suite des formulaires de saisie dans le formulaire Navigation créé aux chapitres précédents.

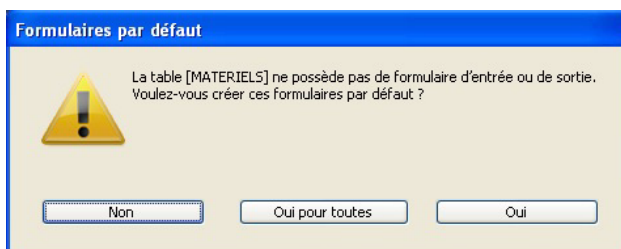
MISE EN OEUVRE

Pour cette première saisie, nous utilisons les formulaires standard, créés automatiquement par 4D :

- 1 Choisissez "Afficher la table courante" dans le menu "Enregistrements".

4D vous propose de créer les formulaires.

- 2 Cliquez sur "Oui pour toutes".



4D a créé 2 formulaires :

- le formulaire ENTREE qui présente l'ensemble des champs d'un enregistrement.
- le formulaire SORTIE qui présente la liste des enregistrements de la sélection.

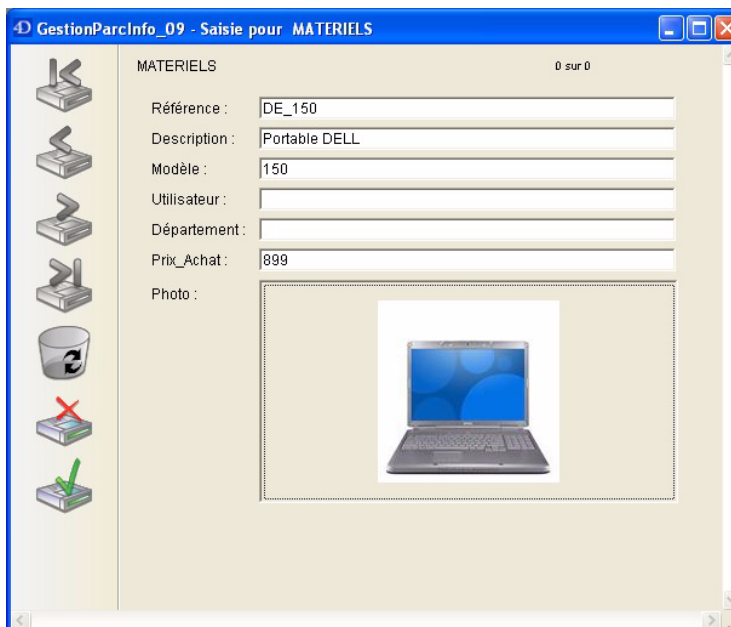
3 Choisissez la table "TECHNICIENS" dans la liste des tables.



4 Choisissez "Nouvel enregistrement" dans le menu "Enregistrements" :




5 Saisissez les données comme sur l'exemple ci-dessous.




Pour passer d'un champ à l'autre, vous pouvez utiliser la touche **Tabulation**.

Pour valoriser le champ image, vous disposez de 3 possibilités :

- Copier une image et la coller dans le champ
- Glisser l'image à partir d'une autre application ou d'une fenêtre système
- Utiliser la fonction d'import du menu contextuel de l'image.

6 Pour enregistrer la fiche, cliquez sur le bouton "valider"  .
4D propose par défaut de saisir un nouvel enregistrement.

7 Pour arrêter et revenir à la liste, cliquez sur le bouton "Annuler"  .

Pour réafficher le détail d'un enregistrement, vous pouvez double-cliquer dessus (Attention, cela ne fonctionne que si l'enregistrement n'est pas en cours de saisie dans la liste).

Pour saisir des informations dans une autre table vous pouvez au choix :

- Cliquer sur le nom de la table concernée dans la liste des tables.



(Pour réafficher la liste des tables ci-dessus, demandez "Liste des tables" dans le menu "Enregistrements")

- Choisir une des dernières tables utilisées via le menu "Enregistrements".

Vous remarquerez au cours de ces saisies que de nombreux éléments peuvent être améliorés (valeurs par défaut, filtres de saisie pour les dates, liste de valeurs proposées...)

La saisie table par table comme nous venons de la faire est pratique, mais ne permet pas, a priori, de contrôler l'existence d'enregistrements dans les autres tables, ni de garantir une quelconque intégrité.

Pour cela, 4D propose 3 solutions :

- l'utilisation des liens,
- la programmation,
- un mélange des deux.

Nous aborderons ces 3 options par la suite en commençant au chapitre suivant par la mise en place des liens.

EXERCICE



Terminez la saisie dans la table TECHNICIENS à partir de la liste d'informations fournies.

Exercice de synthèse :

- Créez la table LOGICIELS à partir des données ci-dessous :

LOGICIELS	
ID	2 ¹⁶
Nom	
Version	
Editeur	
Système	
Prix_Achat	2 ³²

- Créez les formulaires automatiquement,
- Saisissez les 3 premières lignes de données.



Voir corrigé page 239.

COMMENTAIRES



Vous disposez de quatre options pour créer un nouvel enregistrement :

- Cliquer sur le bouton Ajouter => la saisie se fait dans la liste.
- Choisir **Nouvel enregistrement en liste** dans le menu **Enregistrements** => la saisie se fait également dans la liste
- Choisir **Nouvel enregistrement** dans le menu **Enregistrements** => la saisie se fait par l'intermédiaire du formulaire entrée (contenu détaillé d'un seul enregistrement)
- Double-cliquer sur une ligne vide pour afficher un nouvel enregistrement dans le formulaire entrée

Pour aller plus loin

Pour toutes les fonctionnalités relatives à l'utilisation (recherches, tris, états, exports...) je vous conseille de vous référer à la documentation 4D

9

Liens

Objectif(s) pédagogique(s)	<ul style="list-style-type: none"> • Comprendre l'intérêt des liens dans le modèle de données, savoir les tracer et les paramétrer. • Découvrir la puissance des liens 4D, hors programmation.
Durée estimée	10 minutes
Ce que nous allons utiliser	Relations, intégrité automatique

MISE EN OEUVRE

4D est un gestionnaire de base de données *relationnelles*. La relation entre les tables se matérialise par des *liens* entre la clé d'appel (*FOREIGN KEY*) et la clé primaire (*PRIMARY KEY = identifiant unique*).

La puissance des liens dans 4D repose sur plusieurs automatismes qui accroissent considérablement les fonctionnalités, sans avoir à créer la moindre ligne de programme.

Ces automatismes sont toutefois débrayables, entièrement paramétrables et utilisables par programmation.

Pour tracer un lien le principe est simple : Cliquer sur un champ (Clé d'appel = départ) et glisser jusqu'à un autre champ (Clé primaire = arrivée).

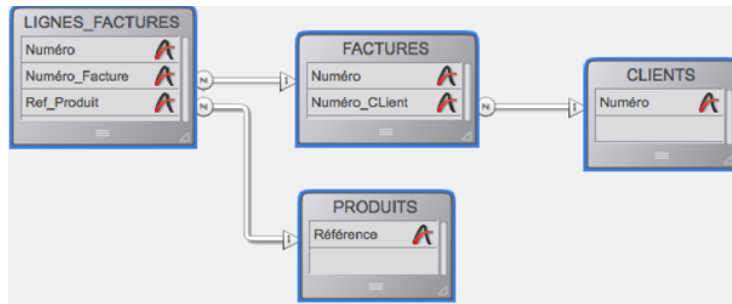
Deux éléments sont à retenir pour tracer un lien :

- les champs doivent être de même type
- on trace toujours le lien d'un champ de la table N (*FOREIGN KEY*) vers un champ de la table 1 (*PRIMARY KEY*).

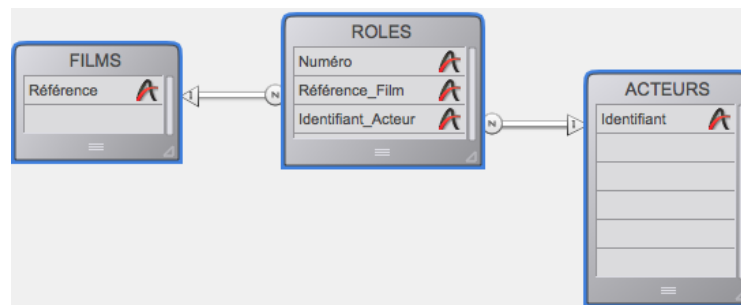
Parlons quelques instants de facturation... je sens que ça vous fait plaisir :=)) On trace les liens :

- de la table Ligne de facture vers Factures (N lignes de facture pour 1 facture)
- de la table Factures vers Clients (N factures pour 1 client)
- de la table Lignes de factures vers Produits (N lignes de factures pour 1 produit)

selon le modèle suivant :



Vous préférez le cinéma ? Soit ! Quelle est la relation entre Acteurs et Films ? Elle passe par l'intermédiaire d'une table "Rôles" sous la forme :



Vous pouvez ainsi associer autant de rôles que vous voulez à un film, et même représenter les différents rôles qu'un même acteur peut jouer dans le même film (Jean Marais dans Fantomas...)

Maintenant que vous maîtrisez les liens : dans quel sens allez-vous tracer le lien entre "Techniciens" et "Interventions" ? Et entre "Interventions" et "Matériels" ?

Voici globalement les questions que vous pouvez vous poser :

- Un même technicien peut-il réaliser plusieurs interventions ?
- Une intervention peut-elle concerner plusieurs Techniciens ?

Et concernant le lien entre "Matériels" et "Interventions" :

- Une intervention peut-elle concerner plusieurs matériels ?
- Un matériel peut-il subir plusieurs Interventions ?

Le sujet peut être débattu longtemps avec moult arguments et exemples... c'est tout l'intérêt de l'analyse et de la confrontation sur le terrain avec les utilisateurs. Dans cet exercice, nous ferons les choix arbitraires suivants :

- une intervention est réalisée par un seul technicien,
- une intervention ne concerne qu'un seul matériel,
- à un matériel sont associés des périphériques, des logiciels et des réseaux.

Nous pouvons donc en déduire que :

- 1 Technicien pourra réaliser N interventions.
- 1 Matériel pourra subir N interventions.

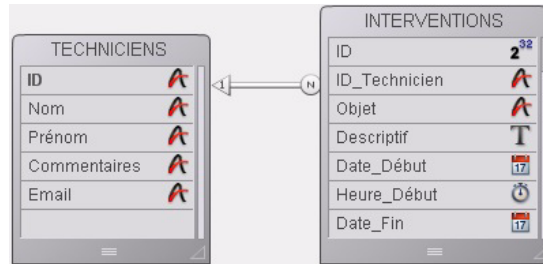
Comme toute base de données relationnelle, 4D utilise des clés "étrangères" qu'il faut faire figurer dans la table N.

1 Si vous ne l'avez pas déjà créé précédemment, ajoutez le champ "ID_Technicien" dans la table Interventions.

Ce champ doit être de même type que le champ clé primaire (ID de la table Techniciens).

Pour créer un lien :

- 2 Tracez le lien entre le champ "ID_Technicien" de la table INTERVENTIONS et le champ "ID" de la table TECHNICIENS.



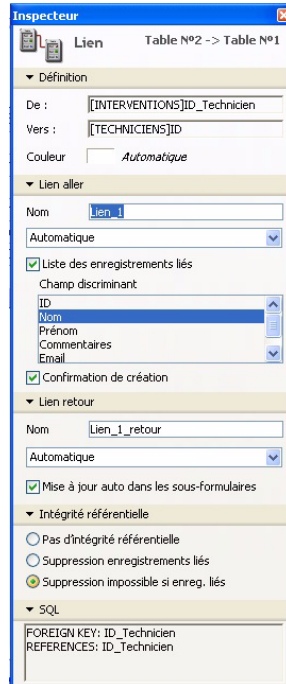
- 3 Double-cliquez sur le lien pour afficher l'inspecteur.

Pour cette première approche, nous souhaitons que 4D travaille à notre place et réalise les jointures (récupération des enregistrements des tables liées) sans programmation, donc automatiquement.

Il faut pour cela paramétrer le lien de la manière suivante :

- 4 Sélectionnez "Automatique" à la place de manuel pour le groupe d'options "lien aller".
- 5 Faites la même chose pour le groupe d'options "Lien retour".
- 6 Cochez également les cases "Liste des enregistrements liés" et "Confirmation de Création".

Elles permettent de gérer une partie de l'intégrité référentielle, nous y reviendrons.



En faisant ces choix, 4D réalisera les requêtes nécessaires pour charger en mémoire les enregistrements liées. Nous pourrons les utiliser à l'affichage par exemple.

Le "lien aller" automatique permet, lors de la saisie d'une intervention, d'avoir en mémoire l'enregistrement du technicien dont on a indiqué le code. Une fois l'enregistrement en mémoire, on peut afficher ses informations à l'écran, imprimer, modifier son enregistrement...

Le "lien retour" automatique permet, lors de la consultation d'un enregistrement technicien, d'avoir en mémoire la liste des interventions réalisées par le technicien et donc de pouvoir afficher la liste sans la moindre programmation.

Nous reviendrons sur les autres paramètres de l'inspecteur de lien un peu plus tard.

EXERCICE



Créez les liens entre les autres tables et paramétrez-les. N'hésitez pas à ajouter des champs, s'il en manque.



Voir corrigé page 239. Notez que dans le corrigé, nous avons déplacé les champs pour que la structure soit linéaire.

COMMENTAIRES



Pour compléter les automatismes des liens, vous pouvez utiliser l'option de contrôle d'intégrité dans l'inspecteur de lien. Elle propose de paramétrer la manière dont 4D doit se comporter lors de la suppression d'un enregistrement de la table contenant la clé primaire :

- **Pas d'intégrité référentielle** : implique une gestion programmée par nos "soins éclairés". 4D ne fait aucun contrôle d'intégrité.
- **Suppression des enregistrements liés** : si on supprime un Technicien on supprime également tous les enregistrements dépendants dans la table N (point de départ du lien). Dans notre cas ce seraient toutes les Interventions.
- **Suppression impossible** : tant que des enregistrements de la table N sont liés à la clé primaire, on ne peut pas supprimer l'enregistrement. Traduit en langage "humain", ça donne : on ne peut supprimer un technicien que s'il n'a réalisé aucune intervention.

Pour aller plus loin

Vous pouvez créer les tables, les champs et les liens par programmation. Consultez la documentation 4D pour découvrir ces possibilités.

Vous pouvez également créer les tables, les champs avec des scripts SQL.

10 Saisie et suppression

Objectif(s) pédagogique(s)	<ul style="list-style-type: none"> • Saisir des données. • Supprimer des données.
Durée estimée	10 minutes
Ce que nous allons utiliser	Mode "manipulation de données", utilisation des liens, suppression

MISE EN OEUVRE

Dans les chapitres déjà réalisés, nous avons créé les enregistrements de Techniciens ainsi que quelques enregistrements Logiciels.

Nous allons maintenant réaliser la saisie en tenant compte des liens mis en place au chapitre précédent.

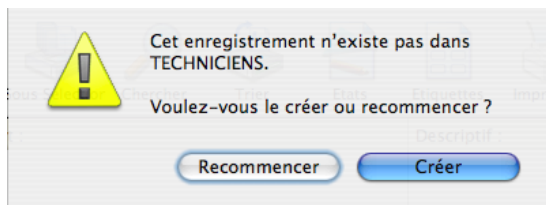
1 Positionnez-vous en saisie sur le premier enregistrement de la table Interventions.

Si 4D vous demande de créer les formulaires manquants cliquez sur "Oui pour toutes".

2 Placez-vous dans le champ ID_Technicien et tapez l'identifiant d'un technicien qui n'existe pas encore.

4D utilise le lien Aller Automatique pour vérifier l'existence de cet identifiant dans la table Techniciens.

Puisque l'enregistrement n'existe pas, 4D nous propose de le créer :



Ce contrôle est dû à la case Confirmation de création que nous avons cochée précédemment. Elle permet de gérer l'intégrité de la base lors de la saisie et éviter ainsi la présence de fiches orphelines.

En cas d'erreur de code, cliquez sur le bouton et ressaisissez un code existant.

Si vous souhaitez créer le technicien avec le code tapé :

- 1 Cliquez sur le bouton .
- 2 Remplissez le formulaire Technicien qui se superpose au formulaire Intervention.
- 3 Validez l'enregistrement.

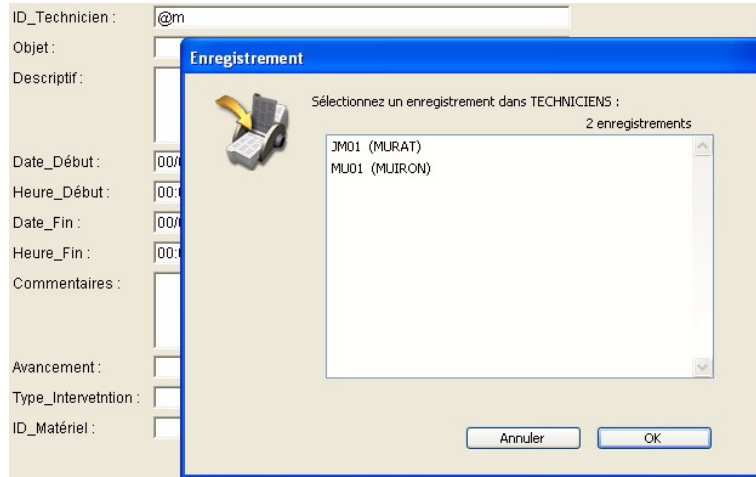
Vous revenez à votre saisie d'Intervention et continuez à remplir l'enregistrement.

Nous venons de réaliser grâce au lien et sans programmation une saisie multi-table avec contrôle d'intégrité.

Nous pouvons encore utiliser le lien pour simplifier la saisie. En effet, nous ne connaissons pas les identifiants des fiches de chacune des tables. Dans ce cas, il suffit de taper le début de l'identifiant suivi du symbole @ pour obtenir à l'écran la liste de toutes les fiches qui "commencent par"...

Vous pouvez également commencer la saisie par @, en tapant par exemple @m. Vous obtiendrez ainsi l'ensemble des techniciens dont l'identifiant contient M, comme vous pouvez le constater sur l'écran

ci-dessous. Le texte qui figure entre parenthèses correspond au champ discriminant sélectionné dans les propriétés du lien.



ASTUCE

Avec les liens, vous avez la possibilité d'utiliser @ n'importe où dans la clé étrangère pour afficher les fiches des clés primaires. Exemples :

- *ab@ = présente toutes les fiches commençant par ab*
- *@ab = présente toutes les fiches finissant par ab*
- *@ab@ = présente toutes les fiches contenant ab*
- *@a@b@ = présente toutes les fiches contenant a et contenant b ("a" d'abord et "b" ensuite)*

La suppression des enregistrements dépend également du paramétrage des liens. Dans ce cas, c'est le "lien retour" qui est pris en compte.

Maintenant que nous avons lié des Interventions à un Technicien, nous allons tenter de supprimer le Technicien (volontairement ou suite à une mauvaise manipulation)... Que doit-il se passer ?


Si le lien retour est manuel		4D n'effectue aucun contrôle et supprime
Si le lien retour est automatique		4D réagit en fonction du paramétrage de l'intégrité
Si	<input checked="" type="radio"/> Pas d'intégrité référentielle	4D n'effectue aucun contrôle et supprime
Si	<input checked="" type="radio"/> Suppression enregistrements liés	4D supprime le technicien et ses interventions
Si	<input checked="" type="radio"/> Suppression impossible si enregistrem...	4D refuse la suppression de la fiche Technicien et ne supprime bien évidemment aucune intervention si le technicien a réalisé des interventions.

Note Les suppressions mentionnées ci-dessus sont conditionnées au mode d'accès aux enregistrements (lecture/écriture) et au verrouillage potentiel des enregistrements par d'autres process utilisateurs.

Ces éléments restent valables quelle que soit la manière dont les enregistrements sont supprimés (utilisation, programmation).

C'est l'analyse des spécifications qui permet de savoir comment paramétrer les liens. Prenons le cas du lien logiciels -> matériels. Doit-on supprimer les enregistrements Logiciels lorsqu'on sort un matériel (rebut, vente, casse, fin de location...)?

A priori, les logiciels ont été achetés indépendamment des machines (sauf OEM) et peuvent être réattribués à un autre matériel. Il semble donc inapproprié de vouloir les supprimer lors de la suppression d'un matériel.

- ▶ Pour supprimer un enregistrement :
 - 1 **Double-cliquez sur le Technicien à supprimer.**
Il s'affiche dans le formulaire de saisie.
 - 2 **Cliquez sur le bouton de suppression**  .
 - 3 **Confirmez le message de suppression.**

EXERCICE



Paramétrez tous vos liens à la lumière des indications fournies ci-dessous :

Création en cascade :

- Créer un logiciel et affectez-lui un matériel qui n'est pas encore enregistré => création de la fiche matériel
- Créer une intervention pour un Matériel inexistant (non encore renseigné dans la base de données) => création de l'enregistrement Matériel. C'est le cas notamment lors de la première installation d'un matériel.
- Indiquez dans l'enregistrement Intervention un Technicien qui n'existe pas (non encore renseigné) => création de l'enregistrement Technicien
- Validez ensuite chacun des enregistrements (ils sont empilés en mémoire)

Vous constatez que sans programmer nous avons pu créer des enregistrements dans 4 tables et assurer l'intégrité des données. Ce principe reste valable si nous travaillons avec 10 tables en cascade.

Pour préparer les imports du chapitre suivant, supprimez tous les enregistrements dans toutes les tables... que se passe-t-il ?

Si vous avez activé la case **Suppression impossible si enregistrem...** sur certains liens, il faut commencer par supprimer les enregistrements dans la table N (point de départ du lien) et remonter de proche en proche.

Par contre, si tous les liens sont avec l'option **Suppression enregistrements liés** cochée, supprimer un matériel supprime en cascade les enregistrements de toutes les tables liées (Interventions, Logiciels...).



Voir corrigé page 239.

COMMENTAIRES



Un grand débat et plusieurs écoles existent sur l'activation/ la désactivation des liens automatiques. Comme toujours quand on a le choix on se pose des questions :=))

D'une manière générale, mes conseils sont :

- Dans tous les cas, tracez les liens.
- Mettez un maximum de liens en automatique si vous devez développer très rapidement une base fonctionnelle.
- Ne mettez aucun lien automatique si vous voulez tout maîtriser et avez peur qu'avec de très gros volumes (plusieurs centaines de milliers, voire de millions d'enregistrements) les automatismes chargent inutilement des enregistrements dont vous n'avez pas besoin
- Quelle que soit la solution que vous choisissiez, il est toujours possible d'activer ou désactiver un lien aller ou retour par programmation.

Pour aller plus loin

La conception d'une structure dépend de nombreux paramètres. je vous conseille de suivre la formation "Concevoir une base de données". Vous y découvrirez les différentes approches, problématiques et solutions envisageables.

11 Importer

Objectif(s) pédagogique(s)	<ul style="list-style-type: none"> • Importer des données. • Créer automatiquement des tables.
Durée estimée	10 minutes
Ce que nous allons utiliser	Mode "manipulation de données", import avec ou sans création de tables

MISE EN OEUVRE

Pour gagner du temps et prévoir la récupération d'un existant informatique déjà saisi, nous allons découvrir les possibilités d'import de données et la création automatique de tables à partir d'un fichier tabulé.

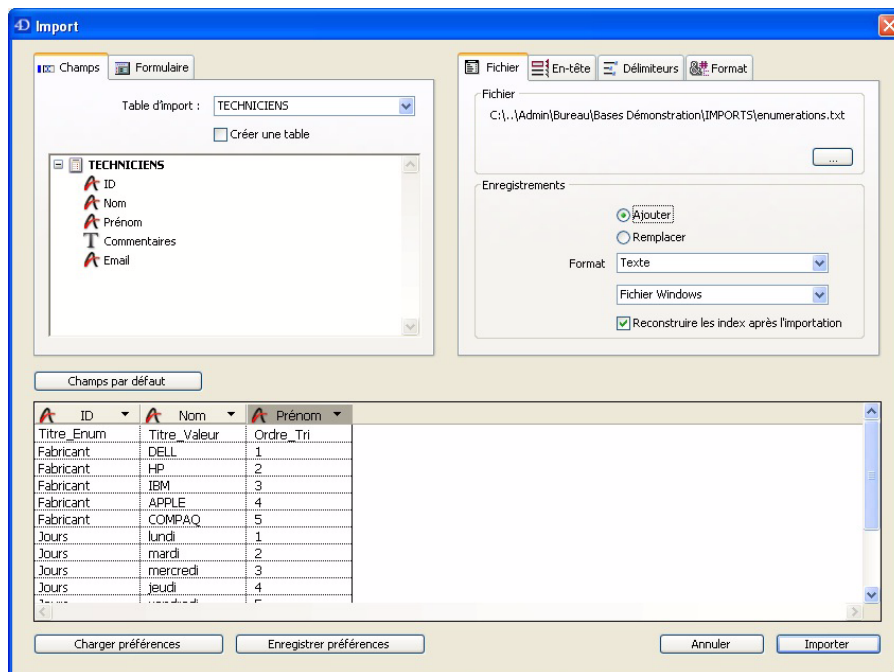
- Pour importer des données :
 - 1 Dans le menu Fichier, choisissez "Importer" puis "du Fichier..."
 - 2 Choisissez le fichier "Techniciens.txt" dans le dossier Bases_Autoformation, sous-dossier Import.
Le dialogue de paramétrages apparaît.
 - 3 Choisissez la table "Techniciens" dans la liste.
 - 4 Faites correspondre chacune des colonnes avec le champ dans lequel elle doit être importée.
 - 5 Cliquez sur Importer.

Les données s'ajoutent aux éventuels enregistrements déjà existants.

Cette fonction d'import permet également de créer les tables directement lors de l'import. Cette fonction est essentielle dans le cas de transfert de données à partir d'autres bases de données ou, comme c'est souvent le cas, à partir de données stockées dans un tableur.

- Pour créer une table lors de l'import :
 - 1 Dans le menu Fichier, choisissez "Importer" puis "du Fichier ..."
 - 2 Choisissez le fichier "Enumerations.txt" dans le dossier Bases_Autoformation, sous-dossier Import.

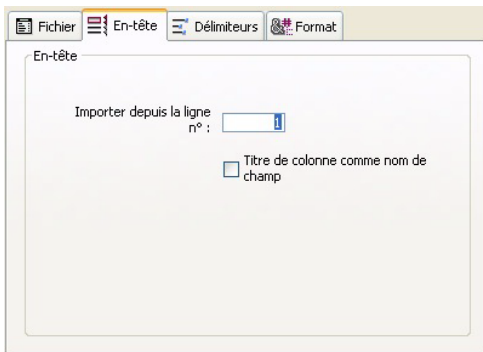
Le dialogue de paramétrage apparaît :



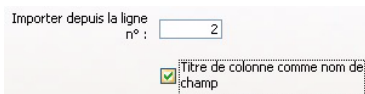
- 3 Cliquez sur la case Créer une table (la liste de tables disparaît et permet la saisie d'un nom).
- 4 Entrez le nom de la table dans la case Table d'import : Enumerations .

Si le fichier que vous importez comporte des titres de colonnes (ce que je vous conseille) :

5 Cliquez sur le 2e onglet de paramétrage :



6 Configurez les paramètres :



Automatiquement la ligne 1 disparaît et tous les champs sont nommés.

Ensuite, que vous ayez sélectionné cette option ou non, vous pouvez définir le type des champs :

7 Cliquez sur chacun des titres de colonne et indiquez le type de champ désiré.

8 Cliquez sur le bouton "Importer".

L'import s'effectue puis 4D propose de créer des formulaires pour cette nouvelle table. L'idée n'est pas mauvaise alors profitons-en !! Cliquez sur **Oui**.

EXERCICE



Importez dans chacune des tables les fichiers proposés dans le sous-dossier Import.



Voir corrigé page 239.

COMMENTAIRES



- Lors de l'import de valeurs booléennes (vrai-faux dans 4D), vous pouvez demander une conversion automatique en allant dans le dernier onglet du dialogue d'import et en précisant "oui;non", "interne;externe", "Homme;Femme"... 4D convertit automatiquement les données importées (la première valeur en Vrai et la deuxième en Faux).
- Vous pouvez également proposer à vos utilisateurs l'accès au dialogue d'import ou d'export en utilisant les commandes
`IMPORTER DONNEES("")`
`EXPORTER DONNEES("")`

D'autres modes d'imports sont disponibles dans 4D :

- Import par formulaires
- Import ODBC
- Importer des données d'une ancienne base
- Connexion à Oracle, MySQL... (utilisation de plug-ins)

Pour aller plus loin

En utilisant ce même dialogue d'import, vous pouvez importer vos données en utilisant un formulaire. Ce formulaire contient les champs devant recevoir les données, il doivent être disposés dans l'ordre des colonnes du fichier d'import. Vous pouvez donc programmer la méthode formulaire et les méthodes objets pour effectuer des contrôles automatiques, des reformatages, le remplissage automatique de champs calculés, etc... comme si l'utilisateur saisisait ses données dans ce formulaire. Vous pouvez également réaliser des imports programmés en utilisant notamment les commandes Ouvrir document et RECEVOIR PAQUET.

12 Formulaires de sortie

Objectif(s) pédagogique(s)	• Utiliser l'assistant de création de formulaires
Durée estimée	15 minutes
Ce que nous allons utiliser	Assistant création de formulaire Sortie + intégration dans page de navigation

MISE EN OEUVRE

Maintenant que la saisie n'a plus de secrets pour vous, nous allons utiliser l'assistant de création de formulaires pour définir les formulaires à intégrer dans l'interface de navigation.

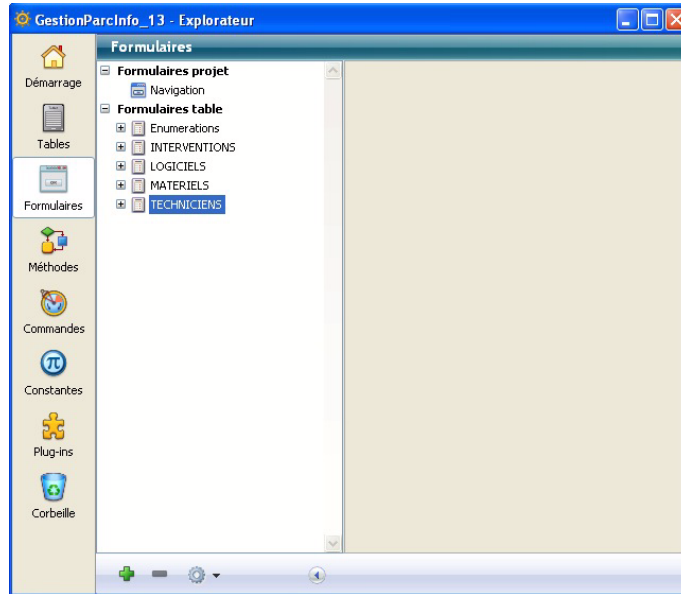
Ces formulaires seront des présentations en liste comme vous l'avez vu sur la base exemple au début de ce guide.



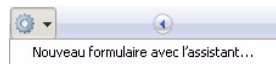
ID :	Nom :	Prénom :	Comm
NB01	BONAPARTE	Napoléon	Source
JM01	MURAT	Joachim	Source
MU01	MUIRON	Jean-Baptiste	Bien st
MA01	MASSENA	André	Source
BE01	BERTHIER	Louis-Alexandre	Source
LA01	LANNES	Jean	Source
NE01	NEY	Michel	Source

- Pour créer des formulaires liste à l'aide de l'assistant :

1 Affichez l'explorateur et choisissez "Formulaires".



- ### 2 Choisissez *Nouveau formulaire avec l'assistant* dans le menu situé au-dessous de la liste :



Vous affichez l'assistant de création d'un formulaire.

- ### 3 Donnez au formulaire le nom : LISTE

- ### 4 Choisissez le type
- Type de formulaire :

- ### 5 Indiquez le modèle de boutons
- Modèle utilisé :

Ils figurent déjà dans le formulaire de navigation.

La table TECHNICIENS est déjà sélectionnée

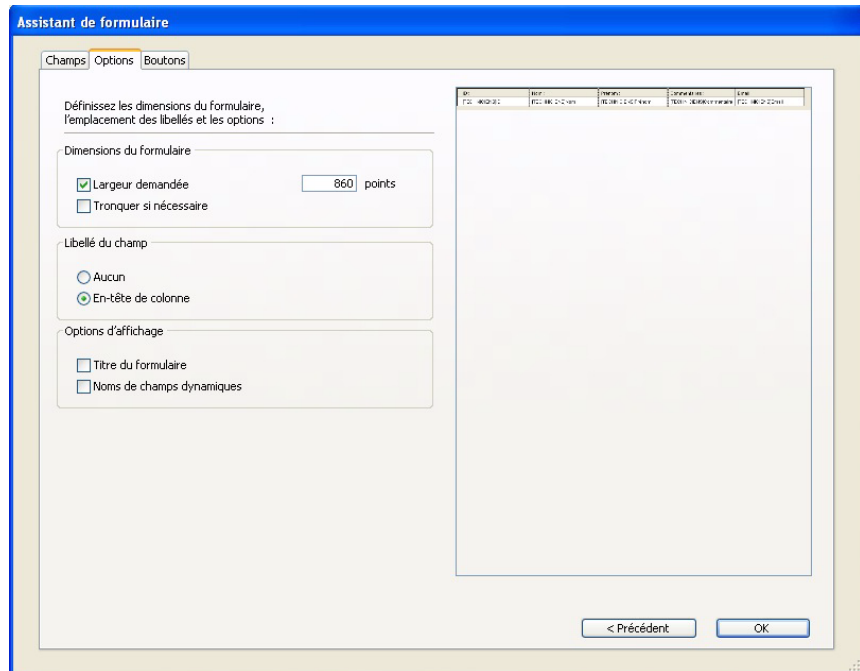
- ### 6 Dans le menu "Liste des champs", choisissez (les champs s'affichent dans l'ordre de création).

- ### 7 Cliquez sur le bouton **»»** pour que tous les champs figurent dans la liste des "champs sélectionnés" et apparaissent dans le formulaire dont on voit une représentation sur la droite.

- ### 8 Supprimez le(s) champ(s) que vous ne souhaitez pas voir figurer en le(s) sélectionnant puis en cliquant sur .

Dans le formulaire Navigation, nous disposons actuellement d'une largeur de 860 pixels pour afficher les listes. Nous devons donc contrôler la largeur des formulaires que nous créons :

- 1 Cliquez sur le bouton **Avancé...**.
- 2 Sélectionnez ensuite "Options" dans l'onglet central.
- 3 Cochez "largeur demandée" et indiquez une largeur de 860 points.



- 4 Cliquez sur le bouton OK pour créer le formulaire.

4D présente l'écran suivant dans lequel nous choisissons de définir un modèle (paramétrez l'écran à l'identique) :

Assistant de formulaire

4D Application peut à présent créer votre formulaire.

Nom du formulaire :

Modèle utilisé :

Modèle

Voulez-vous créer un nouveau modèle en utilisant les paramètres courants ?

Non

Oui

Nom du modèle :



CONSEIL

Un modèle mémorise l'ensemble des paramètres de présentation (sauf les champs) afin de les réappliquer lors de la création de futurs formulaires. En utilisant des modèles, vous gagnerez de nombreux clics de paramétrage dans vos développements importants et garantirez ainsi la cohérence de l'ensemble.

5 Cliquez sur le bouton **Modifier** pour afficher le formulaire en mode Développement.

Formulaire : [TECHNICIENS]LISTE

ID :	Nom :	Prénom :	Commentaires :	Email :
En-tête : 25 [NICIENS]ID	[TECHNICIENS]Nom	[TECHNICIENS]Prénom	[TECHNICIENS]Comment	[TECHNICIENS]Email
Corps : 46	Rupture : 51	Pied : 51		

Vous remarquez la présence d'indicateurs jaunes **En-tête : 25** **Corps : 47** ... Ils précisent la position de chacune des zones du formulaire. Par exemple les objets placés entre l'en-tête et le corps sont affichés 1 fois pour chaque enregistrement de la sélection tandis que ceux placés

entre le haut du formulaire et l'en-tête ne sont affichés qu'une fois en haut du formulaire.

Pour déplacer les taquets, il suffit de les cliquer-glisser.



ASTUCE

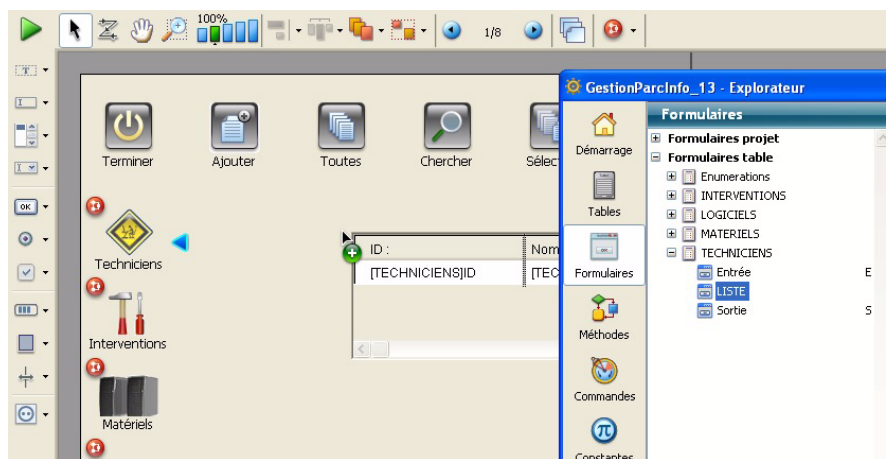
Pour déplacer tous les taquets ensemble, restez appuyé sur la touche Majuscule puis glissez le taquet le plus haut. Vous conservez ainsi le même espacement entre les taquets. La position des taquets peut également être paramétrée dans la liste des propriétés du formulaire.

Taquets	
Entête formulaire	25
Entête formulaire 1	
Corps formulaire	47
Rupture formulaire 1	
Rupture formulaire	52
Pied de Formulaire	52

Nous devons maintenant l'intégrer au formulaire Navigation :

- 1 Refermez le formulaire que nous venons de créer (LISTE).
- 2 Ouvrez le formulaire "Navigation" à partir de l'explorateur.
- 3 Choisissez la page sur laquelle doit s'intégrer la liste des techniciens (page 1 dans notre cas).
- 4 Réaffichez l'explorateur.
- 5 Cliquez sur le nom du formulaire "LISTES" et faites-le glisser sur la fenêtre du formulaire "Navigation".

Le formulaire glissé s'intègre automatiquement dans le formulaire affiché :



- 6 Cliquez dans le formulaire "Navigation"

7 Positionnez le coin supérieur gauche du formulaire glissé en dessous et à droite des boutons existants.

Par défaut le formulaire peut être redimensionné sans contraintes. Pour des raisons ergonomiques il est préférable de le contraindre en largeur et en hauteur :

8 Affichez les propriétés du formulaire (clic-droit sur le fond du formulaire, pas sur un objet).

9 Paramétrez le formulaire comme indiqué ci-dessous :

Sous-formulaire	
Table source	TECHNICIENS
Formulaire liste écran	LISTE
Formulaire détaillé	
Largeur automatique	<input checked="" type="checkbox"/>
Mode de sélection	Multilignes
Saisissable en liste	<input checked="" type="checkbox"/>
Double-clic sur ligne	Modifier enregistrement
Autoriser la suppression	<input checked="" type="checkbox"/>
Sous-formulaire liste	<input checked="" type="checkbox"/>


La largeur automatique permet d'afficher des lignes complètes, ce qui évite qu'elles soient coupées à l'affichage.

Exemple de formulaire contraint et non contraint :

ID :	Description :	Ville :	Téléphone :
site1	bureaux administratifs	Paris	
site2	usineA production	Lyon	
site3	usineb montage	Lille	

Vous savez maintenant afficher les champs des différentes tables. Cependant en l'état, aucun enregistrement n'apparaît bien que vous ayez importé des données dans chaque table au chapitre précédent.

C'est normal car nous n'avons effectué aucune sélection préalable.

Pour sélectionner des enregistrements dans la table Techniciens, il faut juste que vous ajoutiez une ligne de code dans le bouton  du formulaire Navigation :

1	ALLER A PAGE(1)
2	TOUT SELECTIONNER((TECHNICIENS))

- la 1re ligne, nous l'avons vu, permet de naviguer entre les pages.
- la 2e ligne crée une sélection avec l'ensemble des enregistrements de la table (nous reviendrons plus loin sur cette notion de sélection).

EXERCICE



Pour les tables Matériels et Interventions, créez le formulaire d'affichage en liste puis intégrez-le dans le formulaire Navigation.

Pensez à modifier les méthodes des boutons pour réaliser les sélections d'enregistrements.

Pensez également à recopier sur chaque page le pointeur bleu qui indique sur quelle liste on se trouve.



Voir corrigé page 240.

COMMENTAIRES



Si vous souhaitez modifier le modèle que vous avez créé, vous pouvez le modifier de la manière suivante :

- Créer un nouveau formulaire en utilisant le modèle à modifier
- Cliquer sur le bouton "Avancé" pour réaliser les modifications que vous souhaitez apporter.

Allez jusqu'à la création effective de votre nouveau formulaire. Le dernier dialogue vous propose soit de créer un nouveau modèle, soit de modifier celui que vous venez d'utiliser. Choisissez cette 2e option.

Pour aller plus loin


- Utiliser les styles dans les modèles
- Utiliser l'héritage de formulaires

13 Recherches et tris

Objectif(s) pédagogique(s)	<ul style="list-style-type: none"> • Rechercher et trier des enregistrements en fonction de critères d'une ou plusieurs tables • Compléter l'utilisation des liens automatiques
Durée estimée	10 minutes
Ce que nous allons utiliser	Mode "manipulation de données", Editeur de recherche et de tri, programmation

MISE EN OEUVRE

Nous disposons en haut du formulaire de Navigation de la série de boutons que vous avez intégrés dans un précédent chapitre. Nous allons les programmer maintenant en tenant compte de la liste affichée, donc de la page sur laquelle nous nous trouvons.

► Commençons par le bouton "Chercher"  :

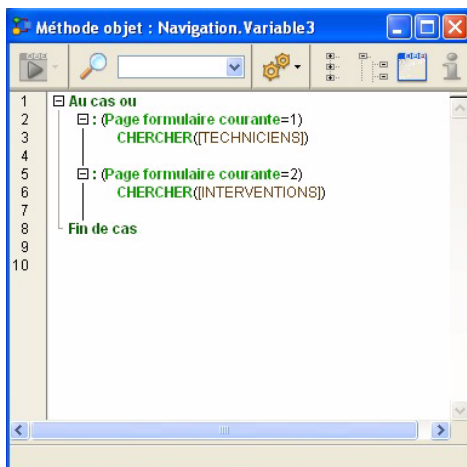
- 1 Affichez le formulaire Navigation.
- 2 Sélectionnez le bouton "Chercher" (sur la page 0).
- 3 Affichez la méthode objet associée (Clic droit -> Méthode objet).

Nous devons utiliser la commande CHERCHER qui prend comme paramètre le nom de la table concernée. Certes, mais quelle sera la table affichée à un instant T ?

Nous savons qu'en page 1 du formulaire nous avons positionné la liste des Techniciens, en page 2 la liste des Interventions, etc.

Il suffit de demander à 4D de nous indiquer quelle est la page affichée et de conditionner la recherche à cette page.

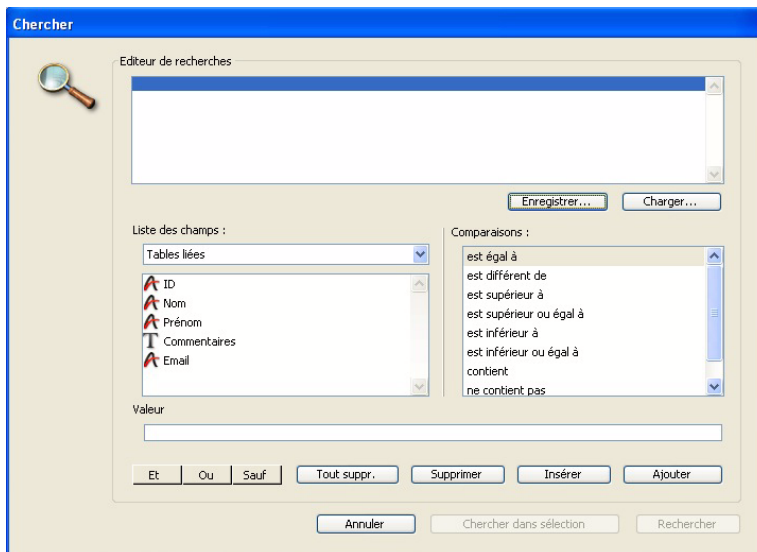
Dans un premier temps, nous allons l'écrire de cette manière :



Note : la fonction "Page formulaire courante" retourne le numéro de page affiché.

- 1 Enregistrez la méthode (Fichier -> Sauvegarder Méthode objet...)
- 2 Exécutez votre formulaire pour tester le fonctionnement de ce bouton.
- 3 Placez-vous sur la liste des Techniciens.
- 4 Cliquez sur le bouton Chercher.

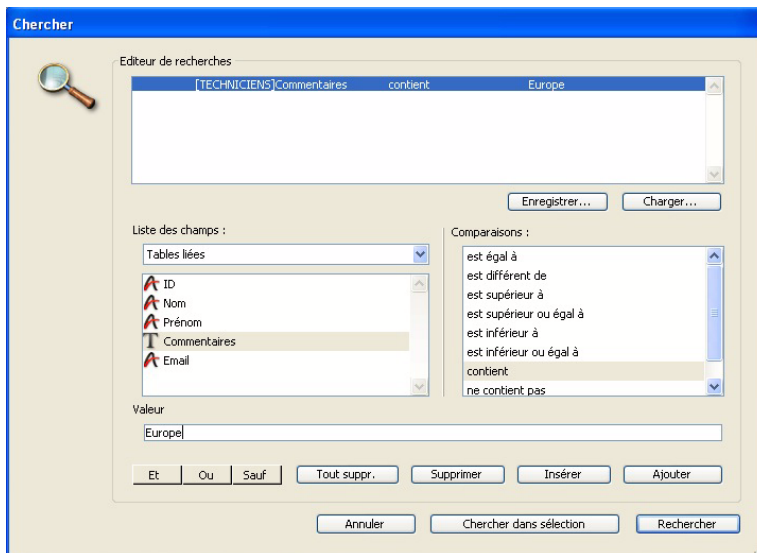
Vous obtenez l'écran de recherche suivant. Les champs correspondent bien à la table Techniciens... Bravo !!! Nous voici sur la bonne voie !



► Pour effectuer une recherche :

- 1 Cliquez sur un des champs puis sur un type de comparaison, puis tapez la valeur recherchée.

La ligne de recherche se complète au fur et à mesure :



- 2 Cliquez sur Rechercher pour lancer la recherche.

4D effectue la recherche demandée et affiche les enregistrements trouvés dans la liste.

Pour cumuler plusieurs critères de recherche, il faut paramétrer la première ligne puis cliquer "Ajouter" et paramétrer les lignes suivantes en les joignant au fur et à mesure par une des conjonctions proposées :

Vous pouvez également effectuer une première recherche puis effectuer une deuxième recherche sur cette première sélection :

1 Procédez de la même manière pour cette deuxième recherche.

2 Lancez la recherche en cliquant sur le bouton .

Nous verrons plus loin comment programmer les recherches les plus souvent effectuées par l'utilisateur.

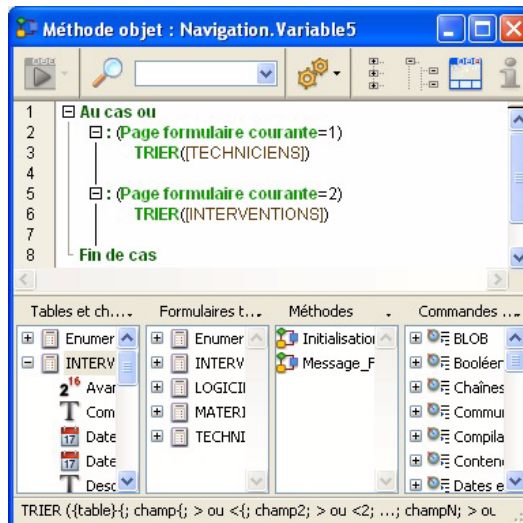
Trier

Le deuxième objectif de ce chapitre est d'apprendre à trier une sélection. Comme pour le bouton de recherche, nous devons adapter le tri à la page affichée. Procédez de la même manière que pour le bouton de recherche :

1 Affichez le formulaire Navigation.

2 Sélectionnez le bouton "Trier" (sur la page 0).

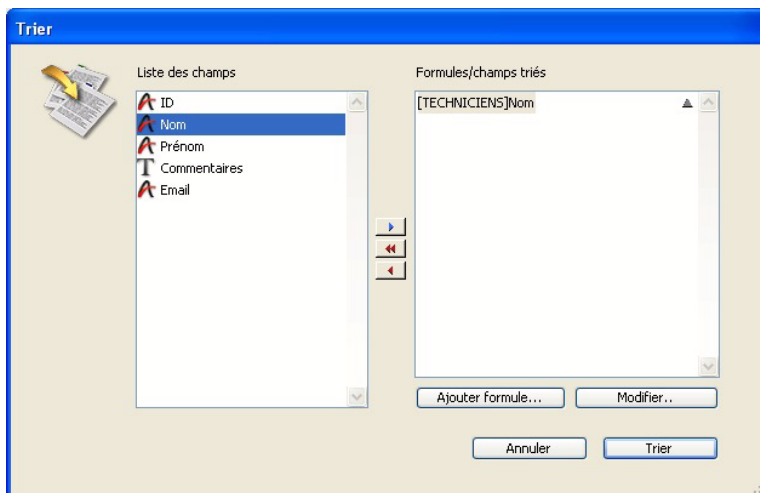
3 Affichez la méthode objet associée (Clic-droit -> Méthode Objet) puis entrez le programme suivant :





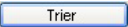
4 Enregistrez la méthode (Fichier -> Sauvegarder Méthode objet...)

- 5 Exécutez votre formulaire pour tester le fonctionnement de ce bouton.
- 6 Placez-vous sur la liste des Techniciens et affichez plusieurs techniciens en faisant une recherche.
- 7 Cliquez sur le bouton Trier.

L'écran suivant apparaît :



Pour définir vos critères :

- 8 Glissez les champs de la colonne gauche vers la colonne droite
- 9 Indiquez ensuite le sens du tri pour chaque colonne en cliquant sur  (Croissant) ou sur  (Décroissant).
- 10 Cliquez sur le bouton Trier  .

EXERCICE



Modifiez la programmation des deux boutons pour qu'ils fonctionnent sur toutes les listes du formulaire de navigation.



Voir corrigé page 240.

COMMENTAIRES



Comme vous l'avez certainement pressenti, il est possible d'optimiser la manière dont nous avons écrit ces méthodes. Nous verrons dans les chapitres suivants comment utiliser la programmation générique, le passage de paramètres entre méthodes, et l'utilisation des pointeurs pour faire référence aux tables, aux champs...

Pour aller plus loin

Vous pouvez compléter la saisie semi-automatique dans l'éditeur de méthodes en modifiant le fichier `Macros.xml` qui se trouve dans le Dossier "Application data" (Windows) "Application Support" (Mac OS X) de l'utilisateur.

14 Etats rapides et export

Objectif(s) pédagogique(s)	• Utiliser l'éditeur d'états rapide pour extraire des données
Durée estimée	30 minutes
Ce que nous allons utiliser	Mode "manipulation de données", Editeur de rapports / paramètres + programmation

L'intérêt de constituer une base de données n'est pas tant d'y entrer des informations... mais plutôt de les restituer !

La restitution s'est effectuée pour l'instant via une visualisation à l'écran dans des formulaires prédéfinis par le programmeur. Or dans de nombreux cas, vos utilisateurs voudront adapter les extractions à des besoins particuliers non définis dans le cahier des charges ou très ponctuels.

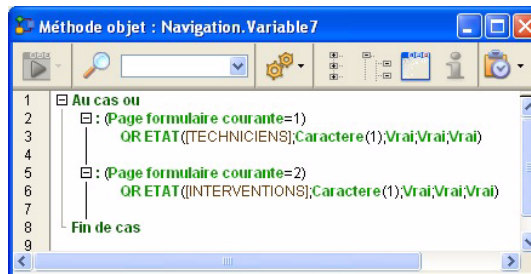
En plus des formulaires prédéfinis par le programmeur et des formulaires modifiables par l'utilisateur (voir documentation 4D), 4D fournit un éditeur d'états paramétrable. C'est ce que nous allons découvrir dans ce chapitre.

MISE EN OEUVRE

L'éditeur d'Etats rapides permet de présenter les données de la sélection courante, c'est-à-dire le résultat d'une recherche préalablement effectuée¹.

1. Le dernier paramètre Booléen de la commande permet d'outrepasser la recherche préalable.

1 Modifiez la méthode du bouton "Etat" du formulaire Navigation :

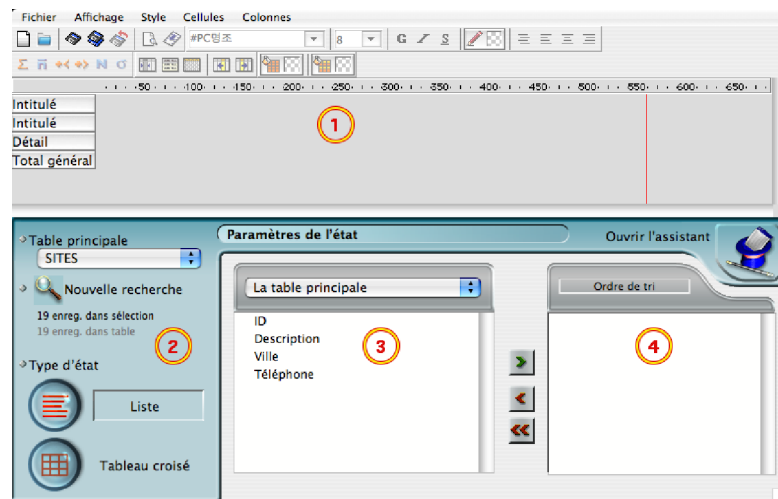


Les paramètres passés à la commande sont :

- le nom de la table : [TECHNICIENS]
- le nom ou chemin d'accès à un document prédéfini. Dans le cas présent Caractere(1) force l'affichage de l'assistant.
- Afficher la navigation hiérarchique entre les tables (utilisation des liens) : Vrai
- Afficher le module d'assistance : Vrai
- Permettre une recherche particulière : Vrai

2 Cliquez sur le bouton "Etat" .

Vous obtenez l'écran suivant :



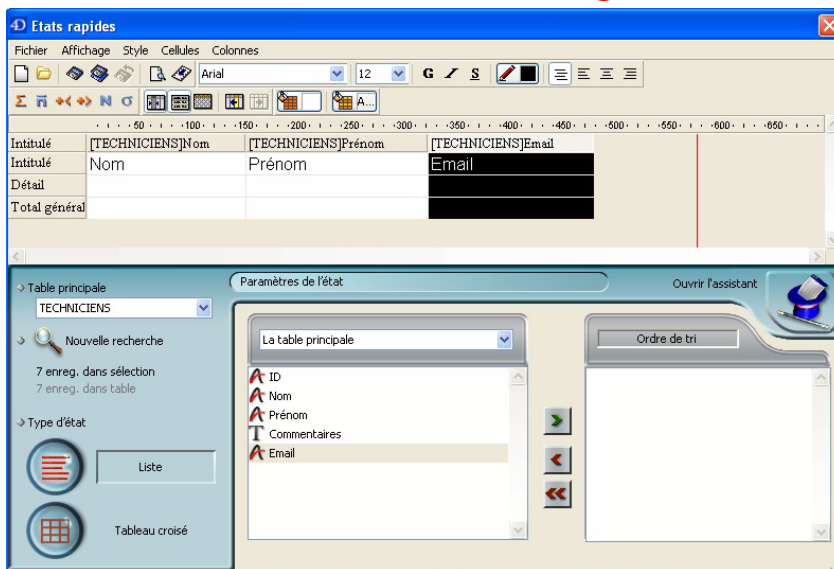
Il se compose de 4 zones :


- 1- La partie haute (barre de menus, barres d'outils et rectangle gris de paramétrage des colonnes)

- 2- La zone sur fond vers qui permet de choisir la table, faire une recherche, définir le type d'état ou ouvrir l'assistant (chapeau de magicien à droite)
 - 3- La colonne de champs de la table sélectionnée
 - 4- La colonne des ordres de tris
- Pour réaliser rapidement un état :

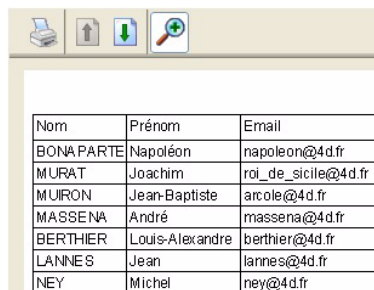
1 Double-cliquez sur les champs de la zone 3.

Ils apparaissent comme colonnes dans la zone 1 :



2 Choisissez "Aperçu" dans le menu Fichier ou cliquez sur l'icône .

Vous voyez apparaître immédiatement la fenêtre suivante :

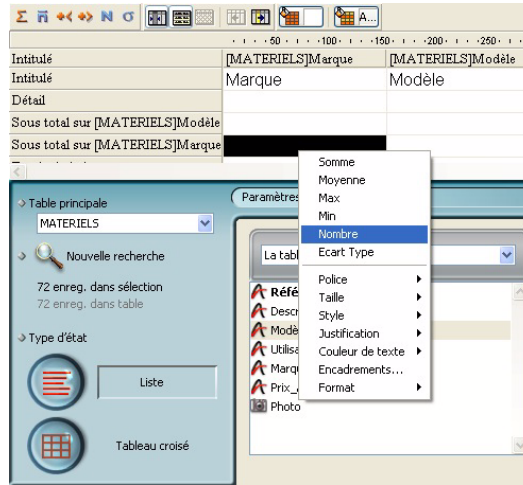


Sur Mac OS, l'aperçu s'affiche en PDF dans l'outil système Aperçu.

3 Pour changer l'ordre de tri, glissez un ou plusieurs champs de la zone 3 vers la zone 4.

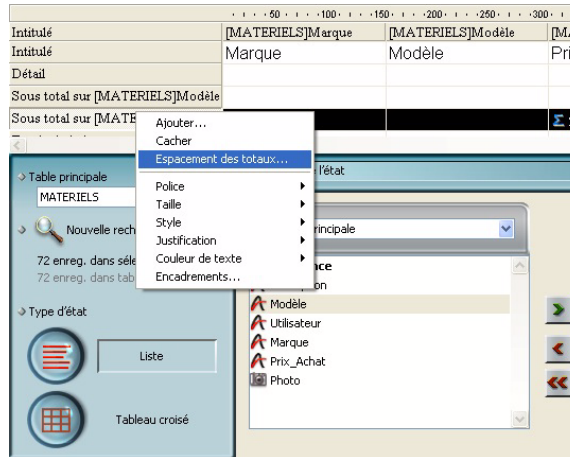
Vous remarquez qu'une ligne de "sous-total" s'affiche dans la zone pour chacun des critères de tri ajoutés :

Dans les lignes de sous-totaux, vous pouvez ajouter des calculs récapitulatifs en faisant un clic droit à l'intersection de la colonne et de la ligne de sous-total :

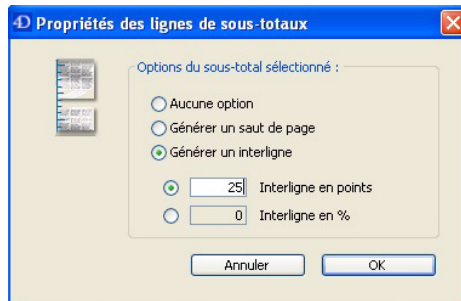


Le calcul réalisé par 4D portera dans cet exemple sur le nombre de techniciens par type d'interventions.

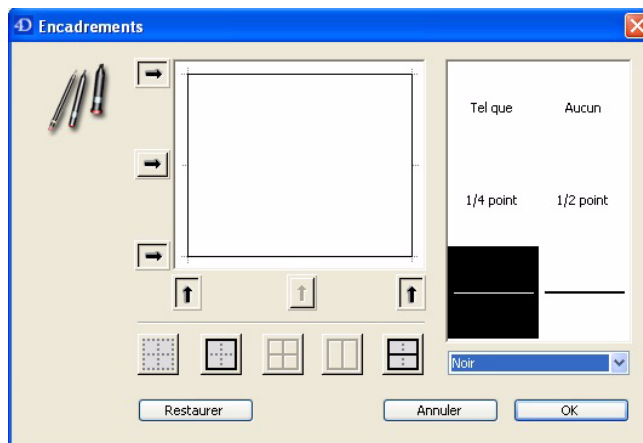
Pour bien distinguer les lignes de détail (chaque enregistrement) des lignes de cumul, vous pouvez les mettre en forme (gras, italique, ...) et gérer des espacements entre le cumul et les lignes suivantes :



4 Demandez un espacement de 25 points...



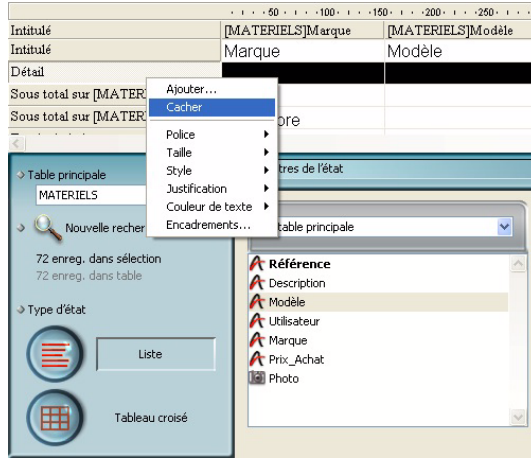
... et un encadrement de la ligne :



Vous obtenez en aperçu le résultat suivant :

Marque	Modèle	Prix_Achat
APPLE	MacBookPro	664.82
APPLE	MacBookPro	49.26
APPLE	MacBookPro	771.57
APPLE	MacBookPro	102.79
APPLE	MacBookPro	815.18
APPLE	MacBookPro	93.33
APPLE	MacBookPro	963.96
APPLE	MacBookPro	200.51
APPLE	MacIntel	101.47
APPLE	MacIntel	673.45
APPLE	MacIntel	723.93
APPLE	MacIntel	480.00
APPLE	MacIntel	583.33
APPLE	MacIntel	438.61
APPLE	MacIntel	608.20
APPLE	MacIntel	683.61
16		7954.02
COMPAG	PRESARIO	12.24
COMPAG	PRESARIO	819.27
COMPAG	PRESARIO	182.74
COMPAG	PRESARIO	704.98
COMPAG	PRESARIO	901.15
COMPAG	PRESARIO	720.91
COMPAG	PRESARIO	446.46
COMPAG	PRESARIO	986.33
COMPAG	PRESARIO	904.39
COMPAG	PRESARIO	976.29
COMPAG	PRESARIO	465.29
COMPAG	PRESARIO	834.96
COMPAG	PRESARIO	863.89
COMPAG	PRESARIO	919.43
COMPAG	PRESARIO	157.35
15		9895.68
DELL	PRECISION	235.18

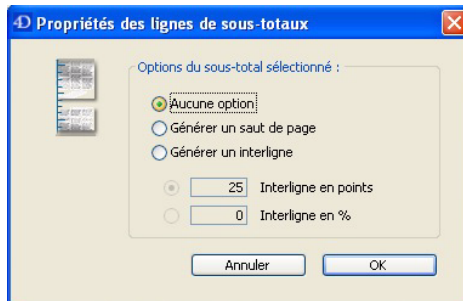
Vous pouvez également n'afficher que les lignes de synthèse en "cachant" le corps (détail de chaque enregistrement) :



Le corps caché est matérialisé par un quadrillage gris sur toute la ligne :

Intitulé	[MATERIELS]Marque	[MATERIELS]Modèle	[MATERIELS]Prix_Achat
Intitulé	Marque	Modèle	Prix_Achat
Détail			
Sous total sur [MATERIELS]Modèle			
Sous total sur [MATERIELS]Marque	Nombre		Σ Somme
Total général			

Dans ce cas nous n'avons plus besoin des 25 points d'espacement, vous pouvez les enlever :



L'aperçu ne présente que les lignes de synthèse :

Marque	Modèle	Prix_Achat
16		7954.02
15		9895.68
16		5847.23
15		7048.29
10		4828.7

Sympa, mais pas très parlant :=)) En effet, nous n'avons pas indiqué à l'éditeur d'états que nous souhaitons savoir à quoi correspondent ces valeurs de synthèse. Pour le savoir il suffit de taper le symbole # dans une des cases de la ligne de synthèse :

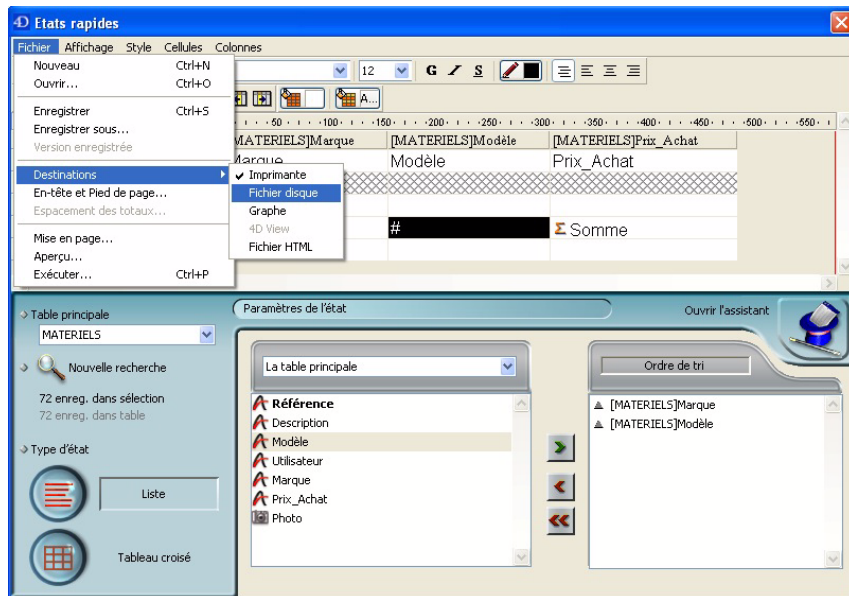
Intitulé	[MATERIELS]Marque	[MATERIELS]Modèle	[MATERIELS]Prix_Achat
Intitulé	Marque	Modèle	Prix_Achat
Détail			
Sous total sur [MATERIELS]Modèle			
Sous total sur [MATERIELS]Marque	N Nombre	#	Σ Somme
Total général			

L'aperçu devient toute de suite plus expressif :

Marque	Modèle	Prix_Achat
16	APPLE	7954.02
15	COMPAQ	9895.68
16	DELL	5847.23
15	HP	7048.29
10	IBM	4828.7

Ce fonctionnement est valable quelle que soit la ligne de sous total dans laquelle vous mettez ces indicateurs. Le résultat calculé ou affiché pour chacun des sous-totaux est évalué par l'éditeur d'Etats en fonction du niveau de regroupement en cours de traitement.

Le résultat de vos états peut être dirigé vers plusieurs destinations :



Pour l'enregistrer sur disque en tant que fichier texte, choisissez "Fichier Disque". Un dialogue système vous permet de donner un nom et de choisir l'emplacement de votre fichier.

Les exports sont également réalisables avec l'Editeur d'Export.



CONSEIL

Usuellement pour un export simple, je vous conseille d'utiliser plutôt l'éditeur d'export. Dès que vos données doivent être synthétisées ou cumulées sous forme de listes ou de tableaux croisés, enregistrées en HTML avec une mise en forme, il est nécessaire d'utiliser l'éditeur d'états rapides.

EXERCICE



Exercez-vous en essayant d'obtenir les états suivants :

- Liste des interventions par technicien
- Nombre d'interventions par technicien
- Nombre d'interventions par type d'intervention et technicien
- Nombre d'interventions par objet et par technicien



Voir corrigé page 241.

COMMENTAIRES



Il est important de bien comprendre et pratiquer l'ensemble des possibilités de l'éditeur de rapports. Vous gagnerez immédiatement en productivité et éviterez de vous casser la tête pour des choses que 4D fait très bien. D'autre part, vous pouvez programmer totalement la génération des états en partant de rapports que vous avez déjà paramétrés comme nous venons de le faire puis en générant le code source correspondant (utiliser l'assistant à l'étape "Finalisation"). En

fait, le générateur d'états est un plug-in intégré à 4D. Il possède son propre langage de programmation.

Pour aller plus loin

- Nous avons rapidement vu comment créer des formulaires avec 4D et y afficher des données. Si on compare ce fonctionnement avec une requête SELECT en SQL :
 - le formulaire contient les colonnes choisies = la première partie du SELECT
 - la recherche et le tri des enregistrements équivalent aux options WHERE et ORDER BY du SELECT
- 4D intègre un moteur SQL et peut fonctionner nativement avec des ordres SQL. En plus de la leçon sur l'utilisation de SQL, référez-vous à la documentation de 4D.
- Programmation de l'Editeur d'Etats (entièrement programmable et paramétrable)

15 Recherche et tri par formule, appliquer une formule

Objectif(s) pédagogique(s)	Recherche avancée et modification en masse des données
Durée estimée	15 minutes
Ce que nous allons utiliser	Mode "manipulation de données", Fonctions avancées, éditeurs de formules

MISE EN OEUVRE

L'ensemble des commandes du langage de 4D est à notre disposition dans plusieurs éditeurs (une version limitée est accessible aux utilisateurs finaux lorsqu'il accèdent aux éditeurs standard).

Cette possibilité permet d'affiner les recherches, les tris, les états rapides... mais également d'appliquer des formules à la sélection d'enregistrements de la table courante, par exemple pour passer un champ tout en majuscules, minuscules, pour concaténer des chaînes de caractères ou au contraire les répartir (si ces traitements n'ont pas été prévus avant l'import, ou si vos formulaires ne précisent pas les conversions et contrôles automatiques).

L'utilisation de formules permet d'effectuer l'équivalent d'une recherche / remplacement de haut niveau. Les commandes utilisées étant celles du langage de 4D, plus vous les connaîtrez, plus vous pourrez progresser dans la finesse des manipulations.

Prenons un exemple : dans le fichier que vous avez importé dans la table Utilisateurs, j'ai volontairement omis de mettre des majuscules aux premières lettres des prénoms et les noms de famille sont mixtes (majuscules, minuscules ou les 2) ; d'autre part, les téléphones ne sont pas tous formatés de la même manière (avec ou sans espace, avec ou sans tirets, 0 non significatif non importé car venant d'une feuille de calcul mal formatée au moment de l'export...). Nous devons donc retraiter ces données erronées ou mal formatées.

4D fonctionne avec la notion de *sélection courante*, c'est à dire une liste d'enregistrements dont les numéros sont conservés en mémoire et qui sert de base à tout traitement (sauf quelques rares cas).

L'ordre d'exécution de toute manipulation se traduit donc par :

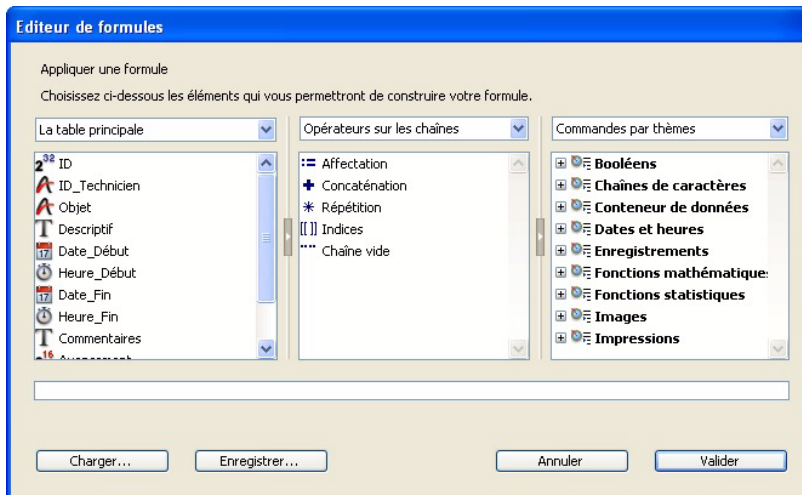
- 1 Sélectionner (créer une sélection d'enregistrements répondant à des critères).**
- 2 Agir (une ou plusieurs actions).**

La première étape consiste à trouver les fiches, soit par une recherche (commande CHERCHER), soit en sélectionnant l'ensemble des enregistrements de la table (commande TOUT SELECTIONNER).

Retraitons d'abord les noms de famille :

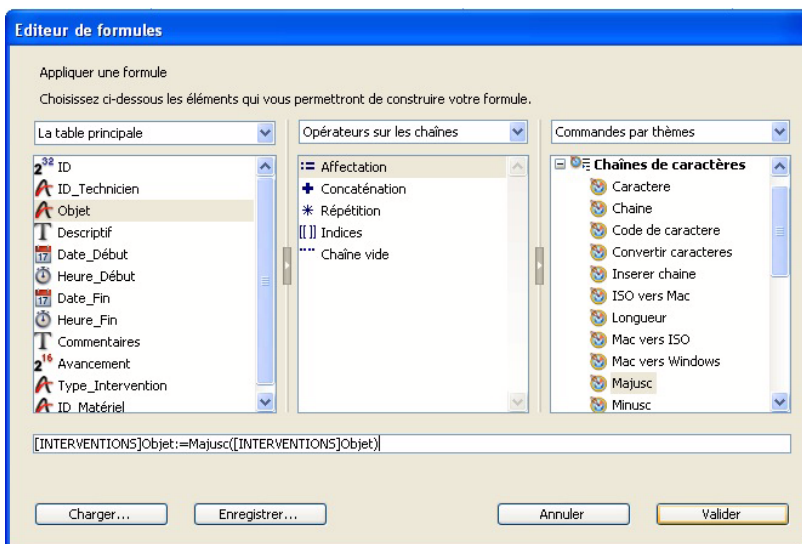
- 1 Sélectionnez la table Interventions dans la liste des tables.**
- 2 Choisissez *Tout montrer* dans le menu *Enregistrements*.**

3 Choisissez "Appliquer une formule" dans le même menu.



Cet écran permet d'écrire la formule dans la ligne en bas, en utilisant le contenu des 3 colonnes. Il suffit de glisser ou double-cliquer sur un nom de champ, un opérateur ou une commande pour qu'il (elle) s'ajoute à la ligne de formule.

4 La formule à taper est la suivante :



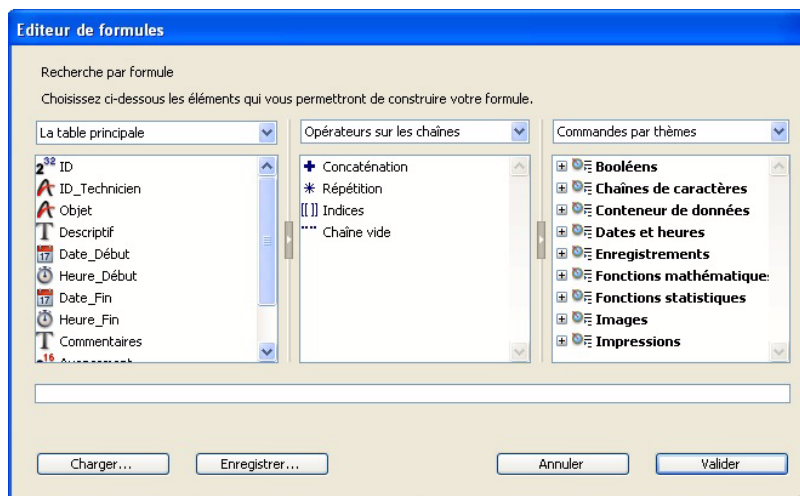
Pour écrire la formule, il suffit de sélectionner par double-clic les valeurs à écrire dans chacune des colonnes.

5 Cliquez sur Valider pour appliquer la formule à la sélection.

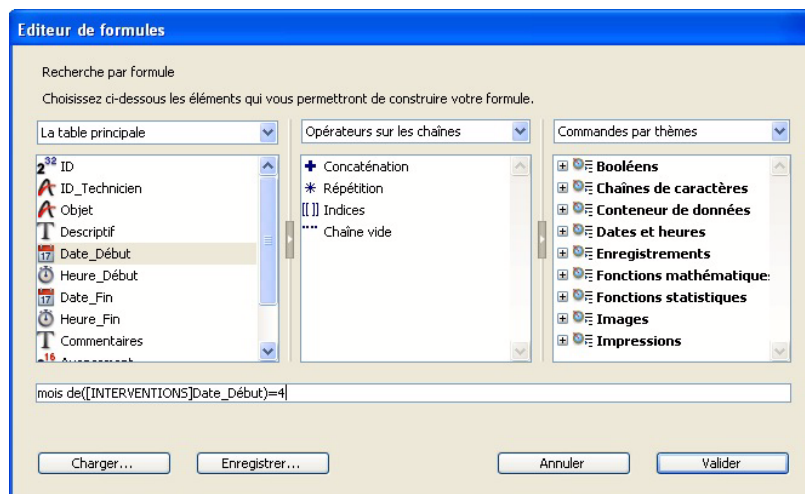


CONSEIL Lorsque vous appliquez une formule, faites le test préalablement sur quelques enregistrements afin de vérifier que votre formule est valide et correspond bien à vos attentes.

La recherche par formule fonctionne sur le même principe, avec le même éditeur de formule, seule la notion d'affectation (:=) disparaît car elle est incompatible avec le contexte de recherche.



On recherche les interventions qui ont eu lieu en avril.



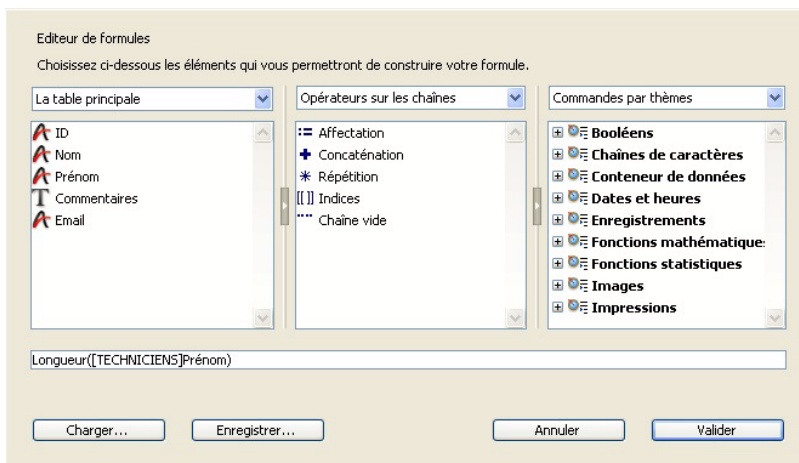
Dernier point : le tri par formule. Il permet d'utiliser de la même manière l'éditeur de formule.

L'exemple le plus parlant est en général le tri d'un dictionnaire qui, par définition, est trié par ordre alphabétique et ne nécessite donc pas de formule. Par contre, qu'en est-il d'un dictionnaire de Scrabble™ ? Il faut d'abord trier les noms par ordre de longueur, en nombre de caractères (mots de 1 caractère, puis 2, puis 3, etc.) puis au sein de ces groupes, trier par ordre alphabétique.

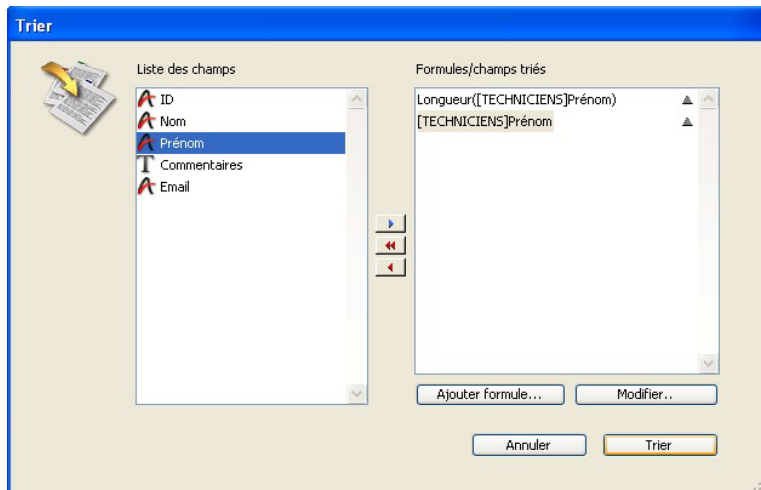
Dans ce cas, vous devez utiliser un tri par formule pour le premier tri.

Dans notre exemple, nous pouvons le faire sur les prénoms des personnes par exemple pour connaître la personne dont le prénom est le plus court (tri croissant) ou le plus long (tri décroissant).

Dans l'éditeur de tri, cliquez sur le bouton **Ajouter formule...** en bas à gauche puis tapez :



Nous pouvons combiner ce critère par formule avec un critère complémentaire en triant pas exemple par ordre alphabétique de prénom :



Nous obtiendrons donc la liste des utilisateurs triée par longueur de prénom croissant et par ordre alphabétique au sein de chaque longueur.

EXERCICE



Exercice 1 : Réécrivez les prénoms avec le 1er caractère en majuscule et le reste du prénom en minuscules.

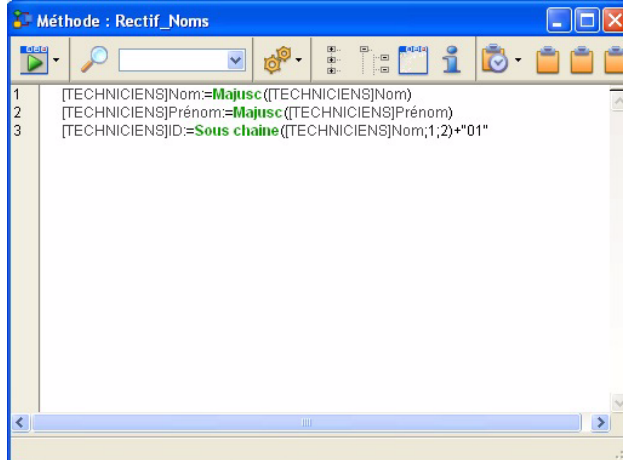


Voir corrigé page 245.

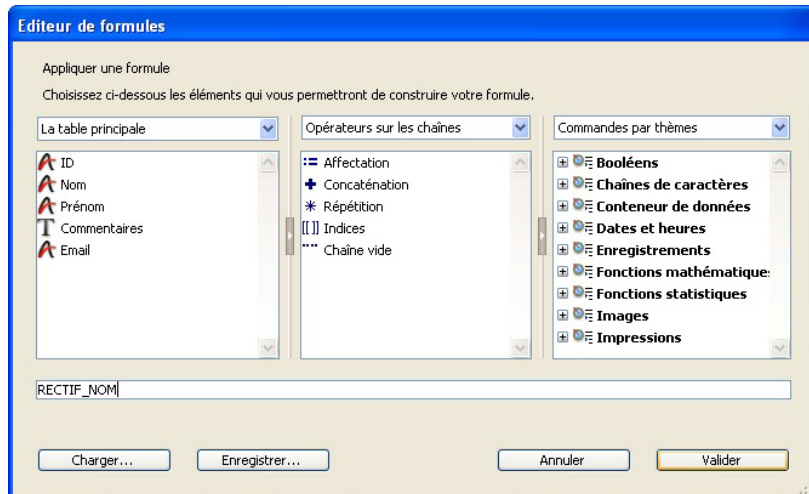
COMMENTAIRES



Vous pouvez également appliquer une méthode à la sélection. Si vous devez appliquer plusieurs formules à une même sélection, vous pouvez écrire au sein d'une méthode les formules à appliquer :



Puis indiquez le nom de cette méthode dans la ligne de formule à appliquer :



Recherche et tri par formule, appliquer une formule

Les formules sont utilisables également dans les colonnes de l'éditeur d'états dont nous avons abordé le fonctionnement dans les précédents chapitres. Cette possibilité vous offre un nombre de combinaisons quasi illimité pour réaliser vos états ou vos exports.

Pour aller plus loin

Pour autoriser/Limiter l'accès aux méthodes dans les éditeurs : référez-vous à la documentation 4D (Préférences)

16

Héritage de formulaires

Objectif(s) pédagogique(s)	Paramétrer les formulaires pour bénéficier d'une interface cohérente, stockée à un seul et unique endroit pour l'ensemble des formulaires.
Durée estimée	10 minutes
Ce que nous allons utiliser	Assistant création de formulaires, palette de propriétés

MISE EN OEUVRE

Jusqu'à présent je vous ai habitué à des manipulations simples... alors continuons :=))

Souvenez-vous de l'utilisation de la page 0 dans les formulaires qui centralise les éléments communs à toutes les pages d'un même formulaire.

L'héritage consiste à créer un formulaire sur lequel on place les objets communs à plusieurs formulaires. Cela permet d'appliquer une modification en une seule fois à un ensemble de formulaires. C'est une sorte de "super page 0".

Dans l'ordre vous devez :

- Créer un formulaire. Nous le qualifierons de "formulaire parent", héritage oblige :=)

- Indiquer le lien de parenté dans les formulaires concernés. Ceux qui héritent : ce seront les "formulaires enfants".

Pour créer le formulaire parent, nous allons créer une table spécifique (INTERFACE) qui contiendra tous les formulaires destinés à des impressions ou affichages particuliers (NB : on ne peut pas hériter d'un formulaire projet).

1 Ajoutez la table Interface.

Nous allons maintenant créer un nouveau formulaire associé à cette table.

2 Affichez l'Explorateur et cliquez sur le bouton "Formulaires".

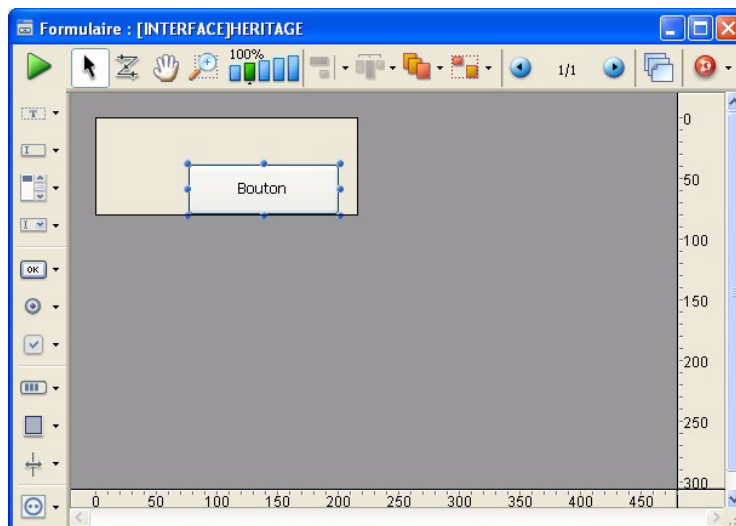
3 Choisissez la table Interface puis cliquez sur le bouton +.

4 Donnez lui le nom HERITAGE puis cliquez sur OK.

Un formulaire vierge apparaît.

5 Tracez un bouton en haut à gauche.

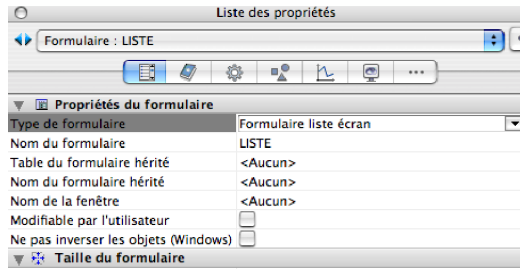
6 Enregistrez votre formulaire (Ctrl+S).



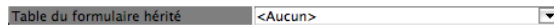
Le parent existe. Nous allons lui associer des enfants :

7 Ouvrez le formulaire "LISTE" de la table TECHNICIENS.

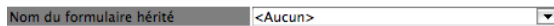
8 Affichez la liste des propriétés :



9 Sélectionner la table Interface dans la propriété



10 Sélectionnez le formulaire HERITAGE dans la propriété



Vous voyez apparaître immédiatement le contenu du formulaire parent en arrière plan.

Si vous modifiez l'emplacement du bouton dans le formulaire parent puis l'enregistrez (sans le fermer), vous voyez la modification se répercuter sur le(s) formulaire(s) enfant(s) ouvert(s). Elle se répercute bien évidemment aussi sur les formulaires enfants fermés, mais c'est moins "visuel" :=))

Détails importants :

- Il n'y a théoriquement pas de limite à l'héritage, un parent peut avoir un parent qui a un parent, mais attention aux histoires de famille et aux conflits de générations. À force d'hériter, les greniers et donc la mémoire se remplissent !!!
- Un formulaire parent est un formulaire à part entière. Il peut donc être constitué de plusieurs pages et dispose d'une page 0. Dans le cadre de l'héritage, seules la page 0 et la page 1 du parent sont visibles (héritables).

Maintenant que vous connaissez tout de l'héritage, vous pouvez le mettre en place.



CONSEIL

Créez au moins un formulaire héritage pour vos formulaires en liste, un autre pour les formulaires de saisie et un dernier pour les formulaires d'informations ou de paramétrage. Les besoins utilisateurs et votre imagination feront le reste...

EXERCICE



Transférez par couper/coller sur un formulaire parent l'ensemble des boutons des formulaires Entrée puis mettez en place l'héritage.



Voir corrigé page 246.

COMMENTAIRES



Dans certains cas, vous aurez besoin qu'un objet du formulaire parent ne soit pas visible ou accessible selon le contexte ou la table concernée.

S'il s'agit d'un bouton, vous pouvez l'inactiver avec la commande **INACTIVER BOUTON**. S'il s'agit d'un objet à masquer, utilisez la commande **CHOIX VISIBLE**.

Vous pouvez également utiliser la commande **DEPLACER OBJET**.

Nous en verrons un cas plus loin dans ce guide lorsque nous optimiserons la programmation avec les méthodes génériques et pointeurs.

17

Formulaires impression

Objectif(s) pédagogique(s)	Créer des formulaires destinés à l'impression, impression en liste et impression détaillée
Durée estimée	15 minutes
Ce que nous allons utiliser	Assistant de création Modifications simples (objets, taquets, propriétés)

MISE EN OEUVRE

Dans 4D, tous les formulaires peuvent être imprimés. On ne distingue pas la notion d'état (impression) de la notion de formulaire (affichage écran).

Dans certains cas particuliers, vous pouvez avoir un même formulaire pour la saisie et pour l'impression. Dans les autres cas, notamment pour des questions de dispositions et tailles différentes, la saisie et l'impression se font à l'aide de formulaires distincts.

Il est fréquent d'avoir plusieurs formulaires pour la même table (vous pouvez en créer jusqu'à 32 000). On indique le formulaire à utiliser soit par programmation soit via les outils d'interface dans le mode Développement.

► Pour créer un formulaire d'impression :

- 1 Utilisez l'assistant comme vous l'avez déjà fait pour un formulaire écran.

2 Dans "Type de formulaire", choisissez selon votre besoin :

Sans
Formulaire détaillé
Formulaire liste écran
Formulaire impression détaillé
Formulaire impression liste

OU

Sans
Formulaire détaillé
Formulaire liste écran
Formulaire impression détaillé
Formulaire impression liste

3 Choisissez ensuite les champs à imprimer :

Création d'un formulaire

Nom du formulaire : Formulaire1

Type de formulaire : Formulaire impression détaillé

Modèle utilisé : XP

Dossier : Niveau supérieur

Table : INTERVENTIONS

Liste des champs :

Tables liées :

- ID
- ID_Technicien
- ID
- Nom
- Prénom
- Commentaires
- Email
- Objet
- Descriptif
- Date_Début
- Heure_Début
- Date_Fin
- Heure_Fin
- Commentaires
- Avancement
- Type_Intervention**
- ID_Matériel

Champs sélectionnés :

- ID
- ID_Technicien
- [TECHNICIENS]Nom
- Objet
- Descriptif
- Date_Début
- Avancement
- Type_Intervention

INTERVENTIONS

ID : [INTERVE:]

ID_Technicien : [INTERVENTIONS]ID_Technicien

Nom : [TECHNICIENS]Nom

Objet : [INTERVENTIONS]Objet

Descriptif : [INTERVENTIONS]Descriptif

Date_Début : [INTERVENT:]

Avancement : [INTERVE:]

Type_Intervention : [INTERVENTIONS]Type_Intervention

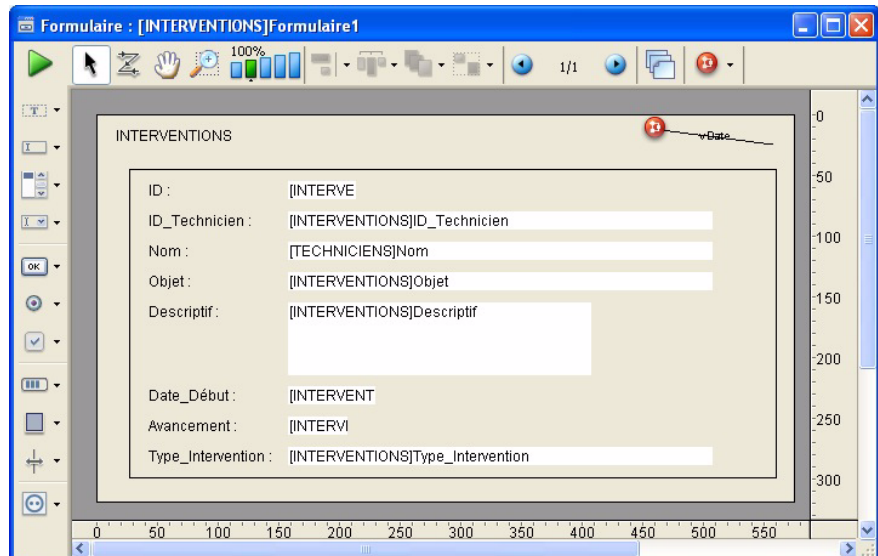
4 Cliquez sur le bouton .

5 Cliquez sur Options afin de régler notamment les paramètres d'impression.

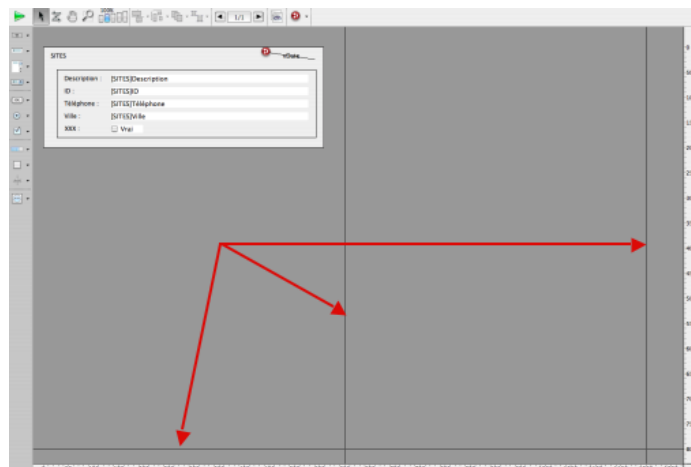
6 Cliquez sur OK.

7 Cliquez sur Modifier (n'enregistrez pas de modèle pour l'instant).

Le formulaire apparaît avec les objets choisis. A ce stade, le formulaire est modifiable à votre guise.

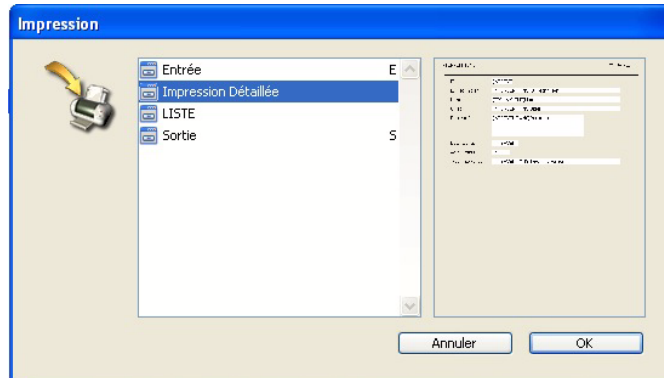


NB : Des traits noirs quadrillent le formulaire et indiquent la taille de la page imprimable du formulaire en fonction des paramètres impression que vous avez définis.



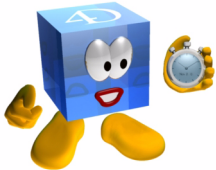
Nous verrons dans les deux chapitres suivants comment améliorer l'apparence et les fonctionnalités de nos formulaires.

- Pour utiliser ce formulaire en impression :
 - 1 Affichez la table INTERVENTIONS à laquelle est rattaché ce nouveau formulaire.
 - 2 Faites une sélection de quelques enregistrements.
 - 3 Choisissez "Imprimer" dans le menu "Fichier".
 - 4 Sélectionnez le formulaire d'impression.



- 5 Cliquez sur OK, les dialogues d'impression apparaissent.

EXERCICE



Mettez en place les formulaires d'impression Liste et les formulaires détaillés sur chacune des tables.

Faites des essais d'impression

Dans un formulaire impression, vous pouvez bien évidemment intégrer un sous-formulaire. Vous allez par exemple créer un formulaire qui permet d'imprimer l'enregistrement technicien ainsi que l'ensemble de ses interventions.

COMMENTAIRES



Attention, l'héritage est réservé à l'affichage. Il n'est pas actif dans le cadre des impressions.

Profitez de ce chapitre pour bien étudier l'ensemble des possibilités offertes par l'assistant de création de formulaires, notamment la partie accessible via le bouton "Avancé".

Pour aller plus loin

Pour compléter cette première approche, je vous conseille de consulter les thèmes ci-dessous dans la documentation de 4D :

- Impression avec rupture
- Impression par zone (Imprimer ligne)
- Dimensionnement des objets lors des impressions
- Déplacement programmé des taquets
- Ajustement automatique de la taille des objets

18 Manipuler les objets sur les formulaires

Objectif(s) pédagogique(s)	Gagner du temps lors des manipulations d'objets
Durée estimée	10 minutes
Ce que nous allons utiliser	Assistant de création Modifications simples (objets, taquets, propriétés)

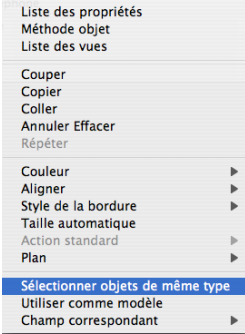
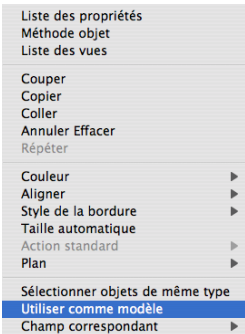
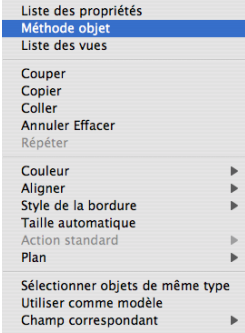
MISE EN OEUVRE

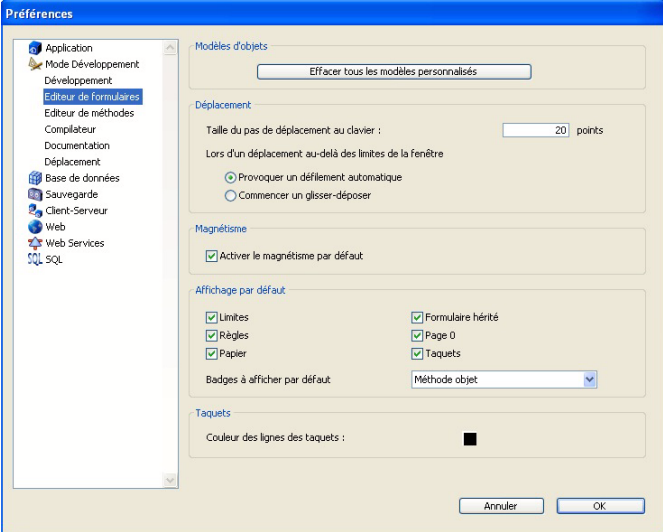
En tant que développeur, vous l'avez certainement constaté, c'est l'ergonomie, l'esthétique et donc la mise en place de l'interface qui est le vecteur le plus chronophage dans un développement.

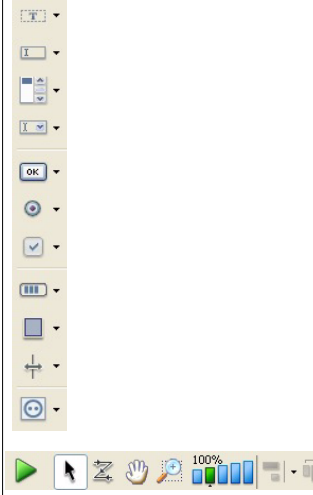
Nous avons déjà utilisé plusieurs moyens qui, d'emblée, vous font gagner du temps. Je souhaite maintenant vous donner les clés de l'optimisation en termes de sélection, déplacement et navigation dans les formulaires.


La juste combinaison clavier-souris est le meilleur moyen d'y parvenir.

Outre les classiques “copier”, “coller”, et autres “tout sélectionner”, voici les indispensables de l’efficacité :

<p>Sélectionner tous les objets de même type d’une même page de formulaire</p>	<p>Clic droit sur un des objets</p> 
<p>Définir les propriétés d’un objet comme devant être le standard pour tous les prochains objets de même type. (NB : on peut définir un modèle pour chaque type d’objet)</p>	<p>Clic-droit sur un objet</p> 
<p>Obtenir la méthode d’un objet. Ce raccourci ouvre la méthode si elle existe ou la crée et l’ouvre.</p>	<p>Alt+clic sur l’objet ou Clic droit sur l’objet +</p> 
<p>Sélectionner / désélectionner plusieurs objets</p>	<p>Clic sur le premier puis Maj-clic sur les autres</p>

<p>Sélectionner avec la souris</p>	<p>Tracer un rectangle de sélection. Tous les objets totalement ou partiellement intégrés au rectangle sont sélectionnés NB : Si vous appuyez sur Alt avant de tracer votre rectangle, seuls les objets totalement encadrés par le rectangle sont sélectionnés</p>
<p>Déplacer un objet d'un pixel</p>	<p>Flèches de direction</p>
<p>Déplacer un objet d'une taille de grille NB : le pas de la grille est défini dans les Préférences</p>	<p>Maj + flèches de direction</p>
	
<p>Agrandir/réduire la taille d'un objet d'un pixel</p>	<p>Ctrl + flèches de direction (Ctrl = touche Commande sur Mac OS)</p>
<p>Agrandir/réduire la taille d'un objet d'une taille de grille</p>	<p>Ctrl + Maj + flèches de direction</p>
<p>Accéder directement à la page 0</p>	<p>Alt + clic sur un objet de la page 0 lorsque vous êtes sur une autre page. Rappel : l'autre utilisation de ce raccourci est d'afficher la méthode de l'objet cliqué.</p>

<p>Accéder aux outils par raccourcis A tout moment, le fait d'appuyer sur la touche...</p>	<p>T = outil Texte F(ield) = Champ ou variable L = ListBox, zones de défilement P = Popup menus, combo box B = Bouton R = bouton Radio C = Case à cocher I = thermomètre, cadrons, S(quare) = rectangle, ligne, ... D = séparateur, onglet X = plug-in</p> <p>Pour la barre horizontale : Z = zoom H = main (Hand)</p> <p>NB : Placez la souris au-dessus des outils pour retrouver ces raccourcis</p>
	

A noter un outil à part : l'ordre de saisie  Il dessine à l'écran l'ordre de focus des objets lorsque l'utilisateur passe d'un champ à l'autre avec la touche **Tabulation**.

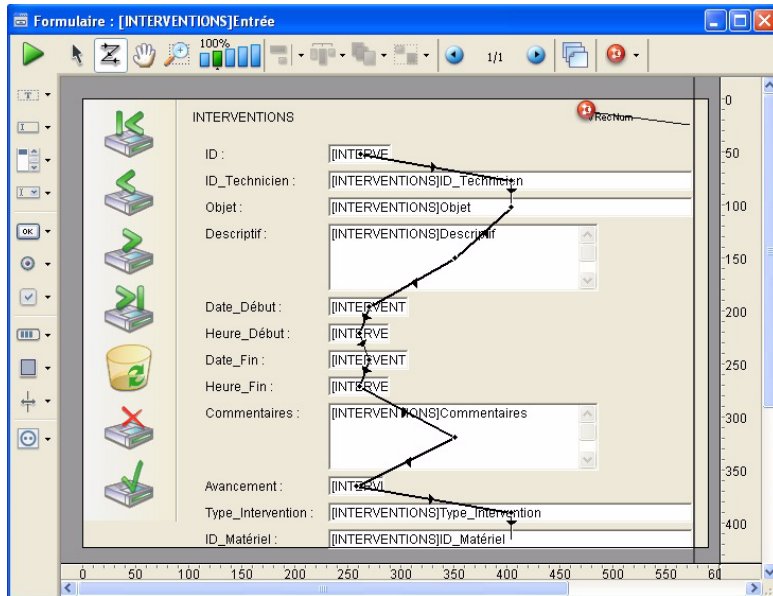
Note : Pour désélectionner cet outil, cliquez sur un autre outil.

Vous pouvez changer cet ordre en cliquant et glissant d'un champ vers un autre... mais ça peut devenir rapidement fastidieux. Le plus simple est de :

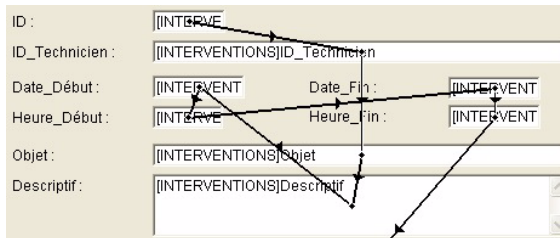
- positionner vos objets,
- tracer un rectangle de sélection autour d'un groupe d'objets (ou tous éventuellement).

4D redéfinit l'ordre de saisie en commençant par l'objet le plus haut puis de gauche à droite et de haut en bas.

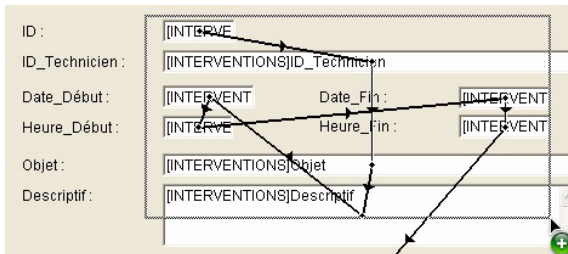
Dans l'exemple suivant, l'ordre de saisie est standard :



Si nous réorganisons le formulaire, l'ordre initial est conservé mais pas très ergonomique :



Tracez un rectangle de sélection en intégrant une partie de chaque objet :



Vous obtenez un ordre de saisie plus cohérent :

The screenshot shows a form with the following fields: ID, ID_Technicien, Date_Début, Date_Fin, Heure_Début, Heure_Fin, Objet, and Descriptif. Arrows indicate a logical flow: from ID to ID_Technicien, then to Date_Début and Date_Fin, then to Heure_Début and Heure_Fin, then to Objet, and finally to Descriptif. This represents a more coherent order of data entry.

Si vous préférez une saisie en colonne, tracez un rectangle sur les deux champs de gauche, puis un autre sur les deux de droite, vous obtiendrez :

The screenshot shows the same form as above, but with two rectangular boxes drawn around the left column of fields (ID, ID_Technicien, Date_Début, Heure_Début) and the right column of fields (Date_Fin, Heure_Fin). This illustrates how to group fields for columnar data entry.

EXERCICE



Mettez en œuvre ces raccourcis, surtout :

- le déplacement et l'agrandissement d'objets
- la sélection avec **Alt**

COMMENTAIRES



Les raccourcis sont un investissement. Perdez du temps en les apprenant pour en gagner beaucoup ensuite.

Gagner du temps et chercher à optimiser peut également se traduire par le glisser-déposer entre deux bases, ou de la bibliothèque d'objets vers un formulaire. Vous gagnerez ainsi de nombreuses manipulations.

Pensez également à la duplication de formulaires, tables et champs.

Pour aller plus loin

- Consultez la liste des raccourcis (fichier PDF dans la documentation de 4D)
- Afficher et modifier la liste des raccourcis (voir le fichier Contents\4D Extensions\4DShortcuts.xml)

19

Propriétés des objets

Objectif(s) pédagogique(s)	Découvrir et paramétrer les principales propriétés des objets sur les formulaires
Durée estimée	10 minutes
Ce que nous allons utiliser	Ajout d'objet et paramétrage de certaines propriétés

MISE EN OEUVRE

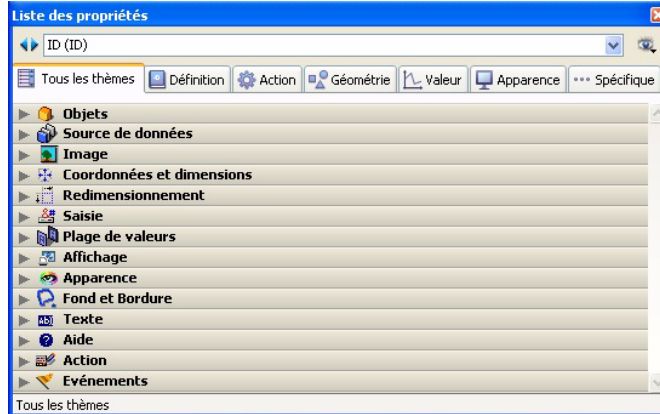
Chaque objet possède des propriétés. Certaines lui sont spécifiques, d'autres sont communes à tous les objets (nom d'objet, coordonnées, hauteur, largeur, redimensionnement...).

Lorsque vous sélectionnez plusieurs objets, seules les propriétés communes apparaissent. Toute modification d'une propriété affecte l'ensemble des objets sélectionnés.

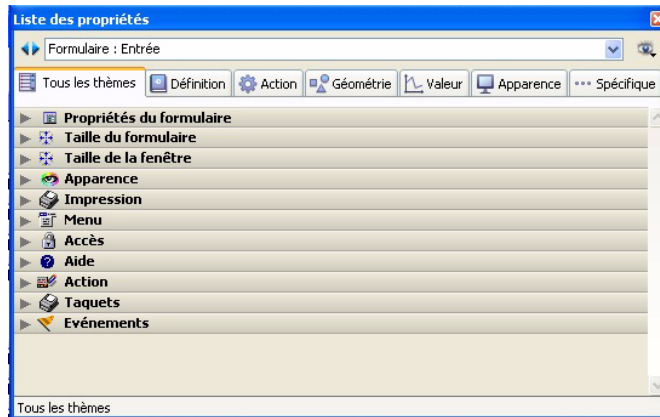
Si aucun objet n'est affiché, vous voyez les propriétés du formulaire.

Propriétés des objets

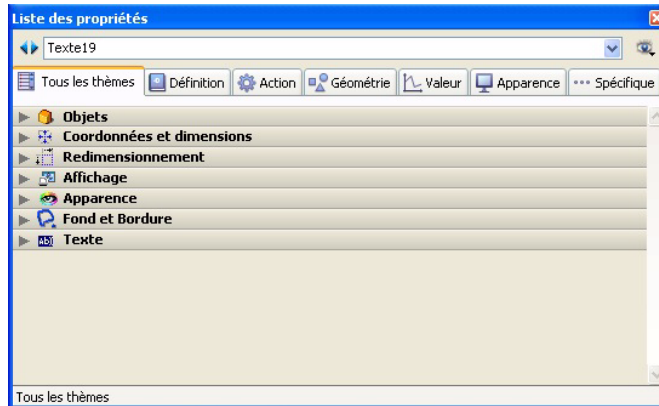
Les propriétés sont classées par thèmes et sont visualisées toutes ensemble ou par thèmes :



Propriétés du formulaire :



Propriétés d'une zone de texte :



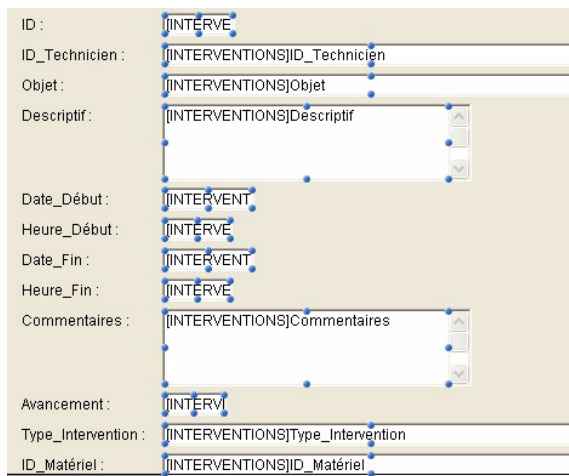
ASTUCE

*Pour ouvrir ou fermer l'ensemble des types, cliquez sur un type en appuyant sur la touche **Ctrl** (Windows) ou **Commande** (Mac OS).*

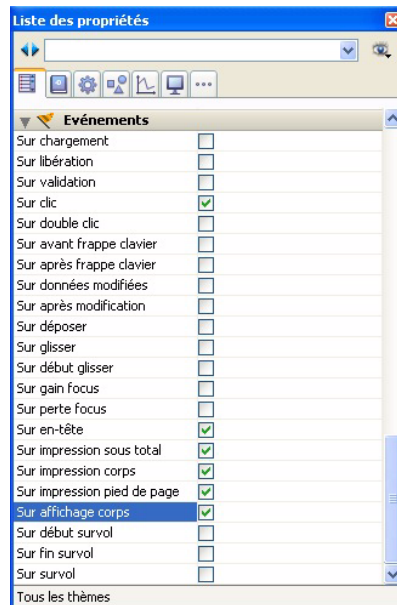
Lorsque la liste des événements est affichée, cliquez sur une case en appuyant simultanément sur la touche **Ctrl** (Windows) ou **Commande** (Mac OS) pour mettre tous les événements dans le même état.

Mise en pratique immédiate : lors de la manipulation de vos formulaires, vous créez ou dupliquez plusieurs objets (des champs ou variables) et souhaitez décocher l'ensemble des événements car aucun ne nécessite l'exécution de sa méthode.

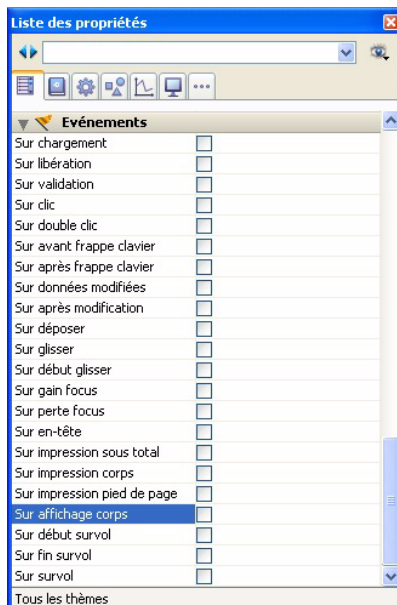
Vous sélectionnez vos objets :



Vous affichez la liste des propriétés :



Vous restez appuyé sur **Ctrl/Commande** puis cliquez sur un seul événement :



Cette manipulation somme toute très simple vous sera très utile si vous convertissez d'anciennes bases (versions précédentes de 4D).

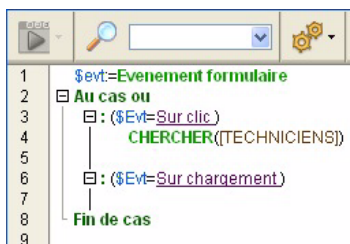
Nous allons tester maintenant quelques propriétés des objets (pour le détail de chaque objet et de ses propriétés, reportez-vous à la documentation complète de 4D).

EXERCICE



Dans le formulaire projet "Navigation", faites en sorte de ne laisser coché que l'événement "Sur clic" sur les boutons de navigation en haut.

- Sélectionnez le bouton *bChercher* puis cochez également la case "Sur chargement".
- Enregistrez le formulaire puis affichez-le en mode utilisateur... *Question 1* : que se passe-t-il ?
- Organisez la programmation de ce bouton comme ceci :



Question 2 : Que se passe-t-il maintenant à l'affichage du formulaire ?



Voir corrigé page 246.

COMMENTAIRES



Par défaut (pour des questions d'optimisation et de facilité de maintenance), je vous conseille de désélectionner l'ensemble des événements objets et formulaire, puis de ne cocher que ceux dont vous avez besoin et auxquels vous avez associé de la programmation. En effet, si vous avez organisé votre programmation avec des tests sur les événements que vous utilisez, la méthode sera quand même chargée et exécutée à chaque fois qu'un des événements cochés surviendra même

si c'est un événement qui ne vous intéresse pas... ce qui sera très fréquent.

Pour aller plus loin

Les formulaires disposent également de propriétés et de méthodes (méthode formulaire). Vous serez donc tenté de placer votre programmation dans l'une ou l'autre de ces méthodes (objet ou formulaire). Mais laquelle choisir ?

Dans l'absolu, ce qui compte c'est que le code soit exécuté lorsque c'est nécessaire... donc l'un ou l'autre des choix conviendra pour débiter. Dans un premier temps, nous ne cherchons pas à optimiser à tout prix.

Retenez quand même l'ordre d'exécution des méthodes lorsqu'un événement survient :

1°) En premier lieu sont exécutées méthodes objets (dans l'ordre de profondeur des objets = niveaux de plan = ordre de saisie)

2°) Une fois toutes les méthodes objets exécutées, c'est la méthode formulaire qui reprend la main.

20

Calculs et formules

Objectif(s) pédagogique(s)	Automatiser les traitements et les calculs
Durée estimée	20 minutes
Ce que nous allons utiliser	Variables, programmation simple

Outre l'enregistrement d'informations "brutes", notre base de données doit nous fournir des informations quantitatives (volumes, prix, totaux) et des calculs à la volée en fonction de nos souhaits.

L'objectif est donc d'automatiser les traitements pour plus de fiabilité. Certaines données devront être conservées et donc stockées dans un champ de la base de données, ce sera le cas par exemple du montant TTC d'un devis. D'autres plus volatiles ne nous serviront que durant la session de travail (elle seront perdues à la fin de la session lorsqu'on quittera 4D). Nous utiliserons des objets de type "variable" dans ce second cas. L'exemple le plus fréquent concerne le nombre d'enregistrements présents dans la liste suite à une recherche.

MISE EN OEUVRE

Pour commencer, nous allons indiquer dans une variable le nombre d'enregistrements trouvés dans la table MATERIELS suite à une recherche par exemple.

La question à se poser en programmant est : "à quel moment doit se déclencher le calcul ?". Dans notre cas, nous considérons qu'il doit se réaliser lorsque l'utilisateur clique sur un des boutons suivants :

- Toutes
- Chercher
- Sélection

Dans tous les autres cas (tri, impression, précédent...), le nombre d'enregistrements ne varie pas, il n'y a donc aucune raison de le recalculer.

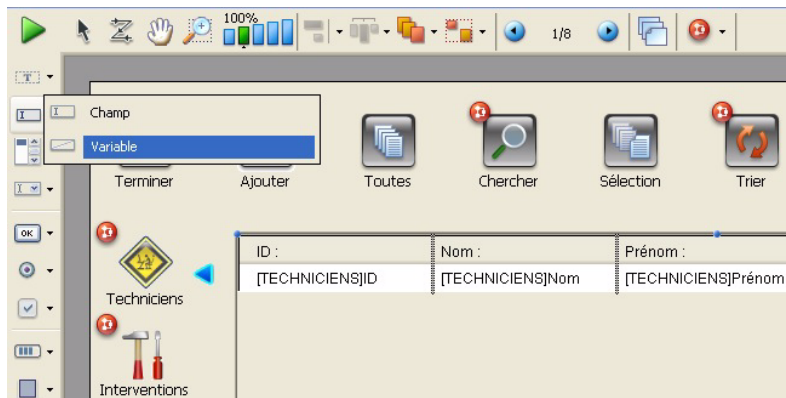
Nous devons ajouter un objet pour visualiser ce nombre d'enregistrements. Alors ?... Champ ou variable ?

Ce sera une variable car nous n'avons aucun intérêt à conserver dans la base de données le nombre d'enregistrements d'une sélection à un instant donné.

Où la placer ? Dans le formulaire navigation certes, mais sur quelle page ?

Sachant que nous souhaitons avoir ce résultat pour chacune des tables affichées, nous placerons cette variable en page 0.

1 Cliquez sur l'objet Champ/Variable :



2 Tracez cet objet juste au-dessous de la ligne des boutons :



La liste des propriétés s'est affichée.

3 Modifiez les propriétés de la manière suivante :

■ nom de la variable :

Objets	
Type	Variable
Nom de l'objet	Variable10
Nom de la variable	vNbEnreg
Type de variable	Alpha

■ non saisissable :

Saisie	
Saisissable	<input type="checkbox"/>
Cacher rectangle de focus	<input type="checkbox"/>
Configuration du clavier	<None>

Les autres propriétés ne nous intéressent pas pour l'instant. Ensuite nous mettons en place la méthode de calcul dans le bouton **Toutes** de la manière suivante :

```

1 TOUT SELECTIONNER((TECHNICIENS))
2 vNbEnreg:=Enregistrements trouvés((TECHNICIENS))
    
```

EXERCICE



Finalisez cette programmation sur les méthodes des boutons **Chercher** et **Toutes**, en tenant compte de la table (donc de la page) sur laquelle on travaille.



Voir corrigé page 247.

COMMENTAIRES



Dans un but pédagogique, nous conservons pour l'instant les méthodes dans les objets. Ultérieurement, nous les déplacerons dans des méthodes plus génériques afin de faciliter la maintenance et la mise en oeuvre de nouvelles fonctionnalités.

Pour aller plus loin

Méthodes objets, méthodes formulaire, trigger et méthodes bases

21

Présentation des variables

Objectif(s) pédagogique(s)	Comprendre les types de variables et leur portée (locales, process, interprocess)
Durée estimée	15 minutes
Ce que nous allons utiliser	<ul style="list-style-type: none">• Les directives de compilation pour définir le typage des variables• La programmation associée à l'utilisation des variables (vie d'une variable : création, utilisation, destruction)

MISE EN OEUVRE

Au cours des leçons précédentes, nous avons utilisé ponctuellement des variables (*vNbEnregs...*). Voyons maintenant en quoi elles consistent, comment elles fonctionnent et, suivant leur définition, quelles sont les limites de leur utilisation.

Si nous comparons le fonctionnement de 4D à une entreprise, nous pouvons en déduire les points suivants :

- Il y a plusieurs services dans l'entreprise qui réalisent chacun une tâche particulière, souvent indépendamment des autres services.
- Chaque service est prévu pour réaliser un certain nombre de tâches dans un ordre donné. Une tâche peut être interrompue car elle dépend du résultat d'un autre traitement.
- Ce nouveau traitement utilisera probablement des informations provenant du traitement précédent, mais également des informations qui lui sont propres.

Si on traduit cela avec un exemple pratique :

- L'entreprise a une usine de production, un service commercial et un service relations humaines.
- Le service Paye centralise les heures réalisées par les autres services et réalise les paies, calcule le nombre de jours de congés, etc.
- Le traitement de la paie implique de connaître le taux des heures supplémentaires ainsi que les différents taux de cotisations. Ces informations sont fournies par le service juridique qui tient à jour la documentation.

Et maintenant faisons le parallèle avec 4D :

- 4D peut gérer plusieurs process (impression, visualisation, palettes d'outils, imports, serveur Web, réponse à des Web services...)
- La méthode qui s'exécute dans chaque process peut comporter plusieurs phases.
- Elle peut faire appel à d'autres méthodes au sein du même process (collègues du même service), ou demander des informations à un autre process (collègue dans un autre service)

Pour chacun des cas nécessaires de communication, nous disposons de variables adaptées :

- Pour disposer d'informations accessibles (en lecture écriture) à tous les process, nous utiliserons des variables **interprocess**. Pour que 4D les considère comme telles, il faut les préfixer par les symboles <> sur Windows et ⋄ sur Macintosh (Ex : <>DateDuJour, <>TableauTaux-Horaires, etc.).
- Durant l'exécution d'une méthode dans un process, elle peut avoir besoin d'une information pour elle seule. Dans ce cas, ce sera une variable **locale**, reconnue dans 4D au symbole \$ qui la préfixe (Ex : \$Compteur, \$ZoneTampon, etc.).
- Toutes les autres variables (les variables **process**, sans préfixe) sont utilisables par plusieurs méthodes dans un même process.

La dernière remarque concernant le besoin d'information provenant du service juridique permet d'introduire la notion de communication entre les process. Il est en effet possible avec 4D de commander la lecture ou l'écriture de variables d'un process vers un autre (et même d'un poste client vers le serveur).

En résumé, dans une école :

- la variable locale est le cahier de l'élève (ou sa feuille de brouillon),

- la variable process est le tableau noir, utilisable et visible par tous les élèves d'une même classe,
- la variable interprocess est le panneau d'affichage à l'entrée de l'école sur lequel figureront les résultats des examens.

La communication interprocess consiste pour un professeur à venir lire ce qui est écrit (ou écrire) sur le tableau de la classe (variable process) de son collègue ou sur le panneau d'affichage de l'école (variable interprocess).

Maintenant que vous maîtrisez la *portée* des variables, nous pouvons en détailler le mode de fonctionnement.

Il existe deux types de variables dans 4D : les **variables simples** (une seule valeur) et les **tableaux** (plusieurs valeurs).

Vous pouvez définir vos variables simples avec mêmes types que les champs (Texte, Entier, Date, Heure, BLOB...) + le type Pointeur. Les tableaux acceptent les mêmes types à l'exception des types BLOB et Heure.

Le cycle de vie d'une variable est le suivant :

Période de la vie	Variable simple	Tableau	Commentaires
Naissance = Initialisation	C_ENTIER LONG(NbJours)	TABLEAU TEXTE(TabDates;0)	En mode Unicode, les types Alpha et Texte sont identiques.
Croissance = Valorisation	NbJours:=25	INSERER LIGNES(TabDates;1) TabDates{1}:=!06/05/2007! OU AJOUTER A TABLEAU(TabDates; !06/05/2007!)	
Compétences au service de la nation "développement" = Utilisation	Boucle(\$i;1;NbJours) Fin de boucle	\$DateDépart:= TabDates{1}+18	Les variables sont en Lecture/Ecriture.
Mort = Effacement et libération de la mémoire	EFFACER VARIABLE(NbJours)	EFFACER VARIABLE(TabDates)	La variable existe toujours, son contenu est réinitialisé.

EXERCICE



Dans une méthode projet que vous nommez “TABLEAUX”, créez 3 tableaux de 26 lignes :

- Le premier contient les lettres de l’alphabet.
- Le deuxième contient les dates à compter d’aujourd’hui.
- Le troisième contient les valeurs de 1 à 26.



Voir corrigé page 248.



CONSEIL

Pour nommer vos variables, prenez l’habitude de respecter une règle afin de vous y retrouver. Soit vous utilisez une nomenclature “dure”, soit vous optez pour des noms de variables “parlants” et donc plus faciles à lire. Pour commencer, je vous conseille des noms clairs et lisibles. Vous pourrez renommer vos variables ultérieurement grâce à la fonction de remplacement global de 4D.

COMMENTAIRES



Comme dans tout langage, les variables sont incontournables dans 4D. Vous pouvez en user et en abuser.

Attention, l’essence même de certaines variables ne permet pas de les visualiser dans un formulaire (tableaux à 2 dimensions, BLOBS...).

Pour aller plus loin

Programmation générique
Pointeurs
Tableaux à 2 dimensions
Blobs

22

Mode trace et débogage

Objectif(s) pédagogique(s)	Maîtriser l'affichage et le fonctionnement du débogueur
Durée estimée	10 minutes
Ce que nous allons utiliser	Débogueur, points d'arrêt

MISE EN OEUVRE

Bien que nous programmions très proprement et sans erreur :=)) il sera parfois nécessaire de vérifier “quand même un peu” l'exécution de notre code, par simple acquis de conscience...

Cette leçon concerne donc l'affichage et le fonctionnement du débogueur.

Pour afficher le débogueur, vous pouvez utiliser un des cinq moyens suivants :

- Insérer la commande TRACE dans le code (et PAS DE TRACE pour arrêter) :

```

4
5
6 TRACE 'DEBUT DU MODE TRACE
7
8 Au cas ou
9   □ : (Page formulaire courante=1)
10      CHERCHER((TECHNICIENS))
11         vNbEnreg:=Enregistrements trouves((TECHNICIENS))
12
13   □ : (Page formulaire courante=2)
14      CHERCHER((INTERVENTIONS))
15         vNbEnreg:=Enregistrements trouves((INTERVENTIONS))
16
17   □ : (Page formulaire courante=3)
18      CHERCHER((MATERIELS))
19         vNbEnreg:=Enregistrements trouves((MATERIELS))
20
21   □ : (Page formulaire courante=4)
22      'PERIPHERIQUES la table n'est pas encore utilisée
23
24   □ : (Page formulaire courante=5)
25      'RESEAUX la table n'est pas encore utilisée
26
27   □ : (Page formulaire courante=6)
28      CHERCHER((LOGICIELS))
29         vNbEnreg:=Enregistrements trouves((LOGICIELS))
30
31 Fin de cas
PAS DE TRACE 'FIN DU MODE TRACE

```

C'est le moyen que vous utiliserez le moins souvent (sauf pour tracer sur le poste serveur).

- Mettre un point d'arrêt : point rouge que vous placez en cliquant dans la colonne des numéros de lignes à l'endroit où vous souhaitez que 4D passe en mode trace :

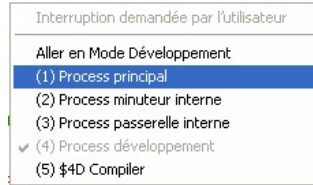
Point d'arrêt — 5 ●

```

6
7 Au cas ou
8   □ : (Page formulaire courante=1)
9      CHERCHER((TECHNICIENS))
10         vNbEnreg:=Enregistrements trouves((TECHNICIENS))
11
12   □ : (Page formulaire courante=2)
13      CHERCHER((INTERVENTIONS))
14         vNbEnreg:=Enregistrements trouves((INTERVENTIONS))
15
16   □ : (Page formulaire courante=3)
17      CHERCHER((MATERIELS))
18         vNbEnreg:=Enregistrements trouves((MATERIELS))
19
20   □ : (Page formulaire courante=4)
21      'PERIPHERIQUES la table n'est pas encore utilisée
22
23   □ : (Page formulaire courante=5)
24      'RESEAUX la table n'est pas encore utilisée
25
26   □ : (Page formulaire courante=6)
27      CHERCHER((LOGICIELS))
28         vNbEnreg:=Enregistrements trouves((LOGICIELS))
29
30 Fin de cas

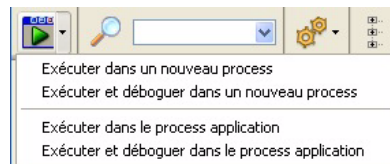
```

- durant l'utilisation de votre programme (en mode application), vous utilisez le raccourci clavier suivant pour choisir le process à passer en mode Trace :
 - Windows : Alt + Maj + clic droit
 - Macintosh : Commande + Alt + Ctrl + clic

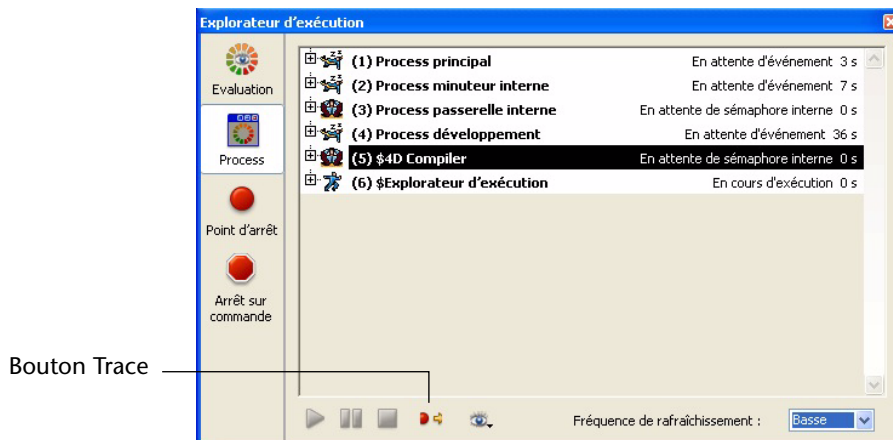


A ce stade des leçons, vous choisirez toujours le Process Principal car nous n'avons pas abordé la création et utilisation d'autres process.

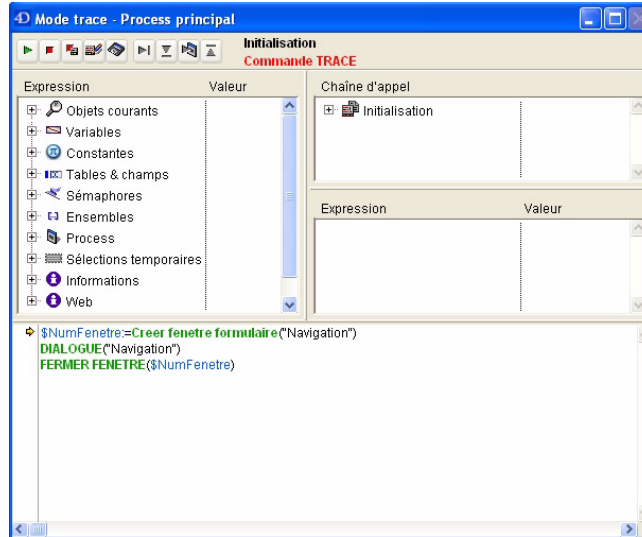
- En mode Développement, cliquer sur le bouton d'exécution de la méthode en choisissant "Exécuter et Débugger".



- le dernier moyen consiste à afficher l'Explorateur d'exécution (Ctrl+Maj+F9) puis à sélectionner le process à tracer et cliquer sur le bouton Tracer :

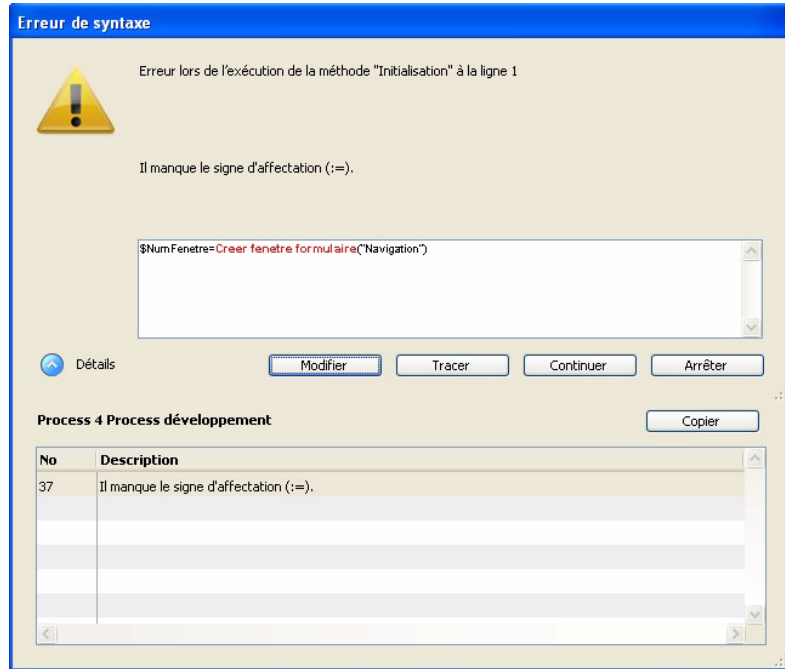


Après avoir effectué un de ces choix, 4D affichera la fenêtre suivante, lors de l'exécution de la prochaine ligne de programme.



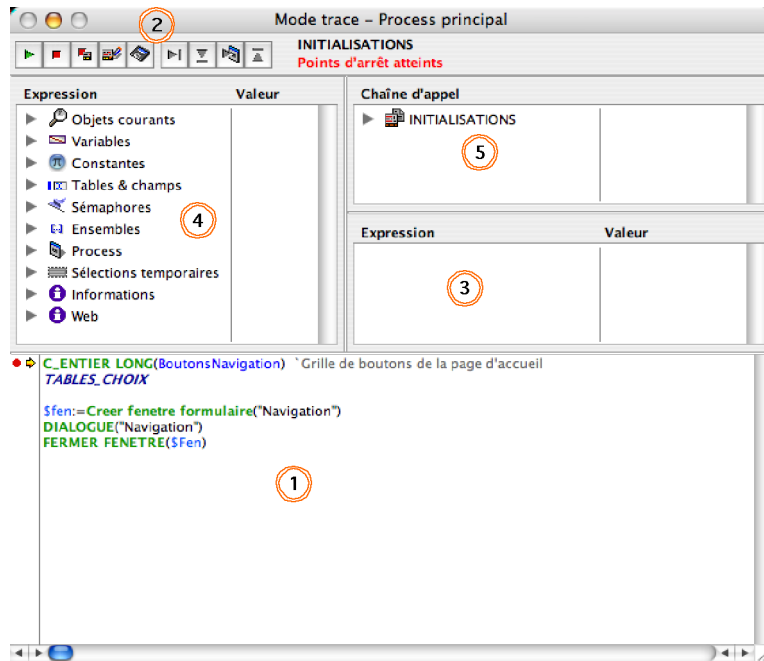
Ces cinq possibilités sont des choix volontaires de passer en mode Trace, mais 4D a aussi son mot à dire ! Lorsqu'il rencontre une erreur de syntaxe, un problème de typage dans une affectation (ou toute autre

erreur) il affiche le message d'information suivant qui permet également de passer en mode Trace :










Dans cet exemple, il manque les symbole : avant le =
nous aurions dû écrire \$NumPage:= \$2

Le débogueur se présente comme ceci :



J'ai numéroté les zones dans l'ordre dans lesquelles vous les utiliserez en général :

<p>La zone de méthode :</p> <p>Vous l'avez compris cette zone présente la méthode en cours d'exécution.</p> <p>Vous pouvez y réaliser plusieurs manipulations :</p> <ul style="list-style-type: none"> - placer/enlever des points d'arrêts - faire glisser vers le haut ou le bas le curseur jaune qui indique la prochaine ligne à exécuter - survoler un champ, une variable ou une table pour voir une infobulle d'information - cliquer sur un objet (attendre 1 seconde en restant appuyé) et le faire glisser dans la zone ③ 	
<p>Les boutons de navigation. En les survolant avec la souris une infobulle vous indique leur fonction :</p>	
<p>Continuer l'exécution en stoppant le mode trace</p>	
<p>Arrêter la méthode</p>	

Arrêter et passer en mode développement pour modifier la méthode	
Rester en mode trace, mais afficher la méthode en mode développeur	
Enregistrer les paramètres (taille de la fenêtre, expressions, largeur des zones, ...)	
Exécuter la ligne en mode "pas à pas" : Si la ligne est un appel de méthode, la méthode s'exécute sans que nous voyions le détail de l'exécution	
Exécuter la ligne en mode "pas à pas détaillé" : Si la ligne est un appel de méthode, on entre dans la méthode et on voit le détail de l'exécution puis on revient à la méthode appelante	
Exécuter la ligne en mode "pas à pas nouveau process"	
Exécuter toute la méthode affichée et revenir en mode trace dans la méthode qui l'a appelée	

EXERCICE



Placez un point d'arrêt dans la méthode base "Sur ouverture" puis redémarrez. Constatez ce qui se passe au fur et à mesure.

Exercices continus lors de la programmation et tests.

COMMENTAIRES



Les deux moyens que vous utiliserez le plus pour passer en mode trace sont :

- le point d'arrêt
- le raccourci clavier

Note : Seuls les utilisateurs référencés comme développeurs (appartenant au groupe qui a accès au mode Développement) peuvent activer le mode trace.

Pour aller plus loin

Avec l'éditeur de débogage vous pourrez réaliser plusieurs autres manipulations ou présentations comme :

- modifier les données des champs et variables en cours d'exécution
- exécuter du code (attention, cela peut être risqué... mais très pratique :=)
- afficher les numéros de tables et de champs
- supprimer toutes les expressions
- choisir les informations qui doivent figurer dans chaque zone de l'éditeur
- mettre des points d'arrêts temporaires, des points d'arrêts conditionnels, etc.

La plupart des possibilités évoquées ci-dessus sont réalisables par l'intermédiaire du menu contextuel (clic droit)

23 Programmation générique (sans pointeurs)

Objectif(s) pédagogique(s)	Optimiser le développement pour plus de clarté, faciliter la maintenance et la portabilité.
Durée estimée	15 minutes
Ce que nous allons utiliser	L'appel de méthodes, le passage de paramètres et la récupération de ces paramètres dans la méthode appelée

MISE EN OEUVRE

Nous avons vu dans les premières leçons de programmation comment paramétrer les boutons en fonction de la page sélectionnée.

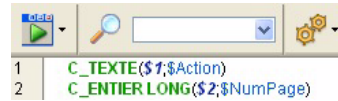
Nous allons maintenant optimiser cette programmation pour ne conserver qu'une seule méthode que tous les boutons de navigation (Chercher, Trier...) vont appeler. Nommons cette méthode *NAVIGATION_FONCTIONS*. Elle va recevoir 2 paramètres :

- Le premier concerne l'action à réaliser sous forme de texte.
- Le deuxième correspondant à la page sur laquelle on se trouve.

L'appel à cette méthode se fait comme ceci pour le bouton "Ajouter" :



Le typage des paramètres s'effectue dans la méthode appelée :

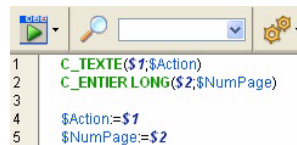


Une méthode peut recevoir jusque 32 paramètres notés \$1 à \$32 (NB : Il est rare d'utiliser plus de 7 ou 8 paramètres.)



CONSEIL

Pour des questions de lisibilité, je vous conseille de créer des variables locales avec des noms explicites auxquelles vous affectez la valeur des paramètres. Il est donc préférable d'écrire :



Le reste de la méthode est une adaptation de ce que nous avons déjà écrit dans les précédents chapitres, à savoir adapter l'action à la

demande et la paramétrer en fonction de la page (donc de la table) choisie :

```

1 C_TEXTE($1;$Action)
2 C_ENTIER LONG($2;$NumPage)
3
4 $Action=$1
5 $NumPage=$2
6
7 Au cas ou
8   : ($Action="Terminer") `Terminer
9     NE PAS VALIDER `action valable quelle que soit la page sélectionnée
10
11   : ($Action="Ajouter") `Ajouter
12     Au cas ou
13       : ($NumPage=1) `Techniciens
14         AJOUTER ENREGISTREMENT ((TECHNICIENS];*)
15         vNbEnreg:=Enregistrements trouvés ((TECHNICIENS))
16
17       : ($NumPage=2) `Interventions
18         AJOUTER ENREGISTREMENT ((INTERVENTIONS];*)
19         vNbEnreg:=Enregistrements trouvés ((INTERVENTIONS))
20
21       : ($NumPage=3) `Matériels
22         AJOUTER ENREGISTREMENT ((MATERIELS];*)
23         vNbEnreg:=Enregistrements trouvés ((MATERIELS))
24
25       : ($NumPage=4) `Périphériques
26         `PERIPHERIQUES la table n'est pas encore utilisée
27
28       : ($NumPage=5) `Réseaux
29         `RESEAUX la table n'est pas encore utilisée
30
31       : ($NumPage=6) `Logiciels
32         AJOUTER ENREGISTREMENT ((LOGICIELS];*)
33         vNbEnreg:=Enregistrements trouvés ((LOGICIELS))
34     Fin de cas
35
36
37
38 Fin de cas
39

```

Ce code n'est pas encore optimisé, mais c'est déjà mieux car nous allons concentrer l'ensemble des actions dans une seule méthode, ce qui simplifie la lecture et donc la maintenance.

EXERCICE



Exercice 1 : Mettez en place ce fonctionnement pour le bouton de recherche et le bouton de tri. Ne le faites pas pour les autres boutons car au chapitre suivant, nous verrons comment rendre l'ensemble encore plus générique en utilisant les pointeurs.

Exercice 2 : Mettez également en place le changement de page à partir des boutons à gauche du formulaire. En créant une méthode *TABLES_NAVIGATION* à laquelle vous passez comme paramètre le numéro de page.

Programmation générique (sans pointeurs)

Ensuite, vous prévoyez la possibilité d'une action en fonction de la page et mettez en place la recherche des énumérations sur la page 8

Les boutons d'ajout, recherche, ... sont inutiles lorsqu'on se trouve sur les pages 7 ou 8. Désactivez-les en fonction de la page affichée.

Pour activer ou désactiver les boutons, vous pouvez le faire un par un ou en groupe. Pour pouvoir l'effectuer en groupe (comme vous le verrez dans le corrigé), il est nécessaire de donner un "nom d'objet" à chacun des boutons.

J'ai choisi de les nommer bNav... (bNavAjouter, bNavChercher, etc.)



Voir corrigé page 249.

COMMENTAIRES



La programmation est un art personnel ! Ne parle-t-on pas juridiquement de "créateur" de programme et de "droits d'auteur" ?

La manière dont ces exercices sont mis en oeuvre vous présente une manière de faire. En fonction de vos habitudes, des normes de développement dans votre entreprise, vous trouverez des organisations différentes, un système de nommage des champs et des variables, un ordre pour passer les paramètres, etc.

Si vous “ressentez” une autre manière de faire, mettez-la en oeuvre et comparez les avantages et inconvénients de chacune des solutions. C’est aussi comme cela que se forge l’expérience.

Pour aller plus loin

Usez et abusez de la programmation générique. La prochaine étape à envisager concerne l’utilisation de pointeurs, c’est-à-dire un complément à ce que nous venons de voir.

La programmation générique vous permet également de diminuer la taille de votre code en concentrant toute une série de fonctions dans une même méthode à laquelle vous passez quelques paramètres textes.

Pour bien mettre en oeuvre les méthodes génériques, retenez ces quelques constations :

- Dès que j’écris pour la deuxième fois des lignes qui se ressemblent... il y a certainement une solution générique à envisager
- (Plus tard) lorsque vous hésitez à créer une méthode générique : créez-la, vous ne le regretterez jamais
- Tout ne peut pas être générique. Parfois, chercher à rendre trop générique engendre des pertes de productivité (perte de lisibilité, difficulté de maintenance).

24 Pointeurs

Objectif(s) pédagogique(s)	Initiation à l'utilisation de pointeurs
Durée estimée	20 minutes
Ce que nous allons utiliser	Concept de pointeur, optimiser la programmation et la rendre générique

MISE EN OEUVRE

Nous l'avons vu à l'exercice précédent, il peut être fastidieux de paramétrer les options de la méthode `NAVIGATION_FONCTIONS` car pour chaque action (chercher, trier, ...), il faut orienter l'action vers la table correspondant à la page sélectionnée.

Dans ce cas, il est plus simple d'indiquer une seule fois au départ la table à traiter. On stocke cette indication dans une variable puis on utilise la variable dans les commandes.

Pour des questions de cohérence, on ne peut pas dire qu'une variable soit égale à une table, ce sont des objets différents à contenu également différent. Nous avons vu qu'une variable peut être de type entier, numérique, texte, image... et donc contenir une valeur du type concerné. Or une table n'est pas un contenu comme le serait par exemple une valeur numérique.

Dans ce cas, on va utiliser des *pointeurs*. L'idée de pointeur est simple et nous l'employons tous les jours dans la conversation courante. Vous en doutez ? Alors vérifions !

Dans la phrase "mon chien est à la maison" nous utilisons 2 pointeurs, c'est à dire 2 indirections. En effet "mon chien" ne permet pas de définir le nom ou la race du chien. Il faut savoir qui parle pour définir de quel chien il s'agit. Idem pour la maison.

Si Paul s'exclame : "range-le dans ce tiroir"... de quel tiroir s'agit-il ? Du tiroir que Paul a désigné avec son doigt. Il a donc pointé un tiroir que l'interlocuteur va "dépointer" (regarder vers quel tiroir pointe le doigt de Paul).

Ce principe simple est à l'oeuvre dans 4D et permet de remplacer les tiroirs par des tables, des champs ou des variables.

Si je demande à 4D de chercher dans cette table, il faudra au préalable que j'aie défini de quelle table il s'agit en la pointant de la manière suivante :



Dans la suite de la méthode, nous allons utiliser le pointeur \$Table dans une phrase ressemblant à "je vais chercher dans la table que tu m'as montrée" ce dernier texte en gras va se traduire comme ceci :

```

4
5 CHERCHER($PointeurTable->)
6

```

Vous voyez qu'il est assez facile d'utiliser un pointeur. L'avantage de cette utilisation est de rendre la programmation encore plus générique et adaptable.

Reprenons l'exemple du chapitre précédent dans lequel vous deviez indiquer pour chaque action la table à traiter en fonction de la page du

formulaire. Nous pouvons maintenant écrire la méthode de la manière suivante :

```

1  C_TEXTE($1;$Action)
2  C_ENTIER LONG($2;$NumPage)
3
4  $Action:=$1
5  $NumPage:=$2
6
7  C_POINTEUR($PointeurTable) `pointeur vers la table correspondant à la page affichée
8
9  `on indique ci-dessous "cette" table
10
11  Au cas ou
12  | ($NumPage=1) `Techniciens
13  | $PointeurTable:=>[TECHNICIENS]
14  |
15  | Fin de cas
16
17  Au cas ou
18  | ($Action="Terminer") `Terminer
19  | NE PAS VALIDER `action valable quelle que soit la page sélectionnée
20  |
21  | ($Action="Ajouter") `Ajouter
22  | AJOUTER ENREGISTREMENT($PointeurTable->*)
23  | vNbEnreg:=Enregistrements trouves($PointeurTable->)

```

EXERCICES



Exercice 1 : Finissez la programmation de cette méthode en remplaçant dans les commandes la [table] par le pointeur vers la table.

Exercice 2 : Pour compléter ce que nous venons de voir et réviser des notions importantes, vous allez modifier le formulaire d'héritage Entrée de la manière suivante :



Vous devrez certainement retoucher un peu vos formulaires entrée. Chacun de ces boutons est disponible en tant que fichier image dans le fichier d'import.

Mettez en place une méthode générique avec pointeurs qui permet de naviguer et de remplir l'ensemble des fonctions.



Voir corrigé page 250.

COMMENTAIRES



Au départ, vous aurez peut-être un peu de mal avec les pointeurs et rapidement, vous constaterez que c'est très simple à mettre en oeuvre et d'une puissance considérable.

Les pointeurs sont très souvent utilisés comme paramètres lors des appels de méthodes. Je vous conseille de passer un peu de temps sur ce thème car il vous servira dès que vous commencerez à vous dire : "je vais dupliquer ce bouton" ou "il faut que je recopie et adapte cette méthode".

Pour aller plus loin

Quand vous devrez gérer des listes de pointeurs, il sera temps de vous intéresser aux tableaux de pointeurs.

Pensez alors également à la commande "Pointeur vers" qui vous permettra de définir le nom de pointeurs à partir d'une chaîne de caractères. Par exemple : `Pointeur vers("MaVariable"+Chaine($i))`

25

Evénements

Objectif(s) pédagogique(s)	Gestion des événements
Durée estimée	35 minutes
Ce que nous allons utiliser	Prise en compte et programmation des événements

Décrochez-vous le téléphone lorsqu'il ne sonne pas ?

Décrochez-vous le téléphone dans le bureau d'un collègue s'il ne vous y a pas invité ?

En général non, bien évidemment. 4D respecte également ces principes. C'est-à-dire que les différentes méthodes ne seront exécutées que si certains événements prédéfinis surviennent. Notre mission est donc de définir les situations et les conduites à tenir, un peu comme on indique à un enfant de ne traverser la rue que si le feu est vert et de n'ouvrir la porte qu'à des personnes connues.

Devenons les contrôleurs aériens de 4D !

MISE EN OEUVRE

Distinguons dans un premier temps les événements "formulaire" des événements "moteur".

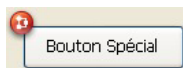
Les événements formulaires ne se déclenchent que si un formulaire est utilisé (à l'écran ou à l'impression), donc lorsque l'interface est manipulée par un utilisateur (clic, glisser-déposer, survol, sélection d'un menu, frappe clavier...)

Les événements moteur ne concernent que les quatre actions sur les données : **création**, **modification**, **suppression** et **chargement**. Ces quatre événements sont gérés par le moteur de données. Cette

remarque est importante car en Client/Serveur, le moteur de données tourne sur le serveur. Vous ne verrez donc jamais un événement moteur se réaliser sur votre poste client. Vous ne pourrez donc pas le tracer non plus d'un poste client.

Nous reviendrons ultérieurement sur les événements moteurs. Dans un premier temps, nous allons nous amuser un peu en créant un bouton "spécial".

1 Tracez un bouton standard sur la page 7 du formulaire Navigation :



2 Modifiez ses propriétés comme suit :

- Nom : bSpecial
- Seul l'événement "Sur début survol" reste coché

3 Editez sa méthode et saisissez le code suivant :

```

1 | Emplacement de la souris au moment du déclenchement de la méthode
2 | POSITION SOURIS($Souris_H,$Souris_V,$BoutonSouris)
3 |
4 | `A quelle position se trouve l'objet dans la fenêtre
5 | LIRE RECT OBJET(bSpecial,$Obj_G,$Obj_H,$Obj_D,$Obj_B)
6 |
7 | `On calcule le centre de l'objet
8 | $Centre_H=$Obj_G+((($Obj_D-$Obj_G)/2)
9 | $Centre_V=$Obj_H+((($Obj_B-$Obj_H)/2)
10 |
11 | $Decalage=10 `Prévoir un décalage complémentaire de x pixels
12 |
13 | Si ($Souris_H<$Centre_H) `si la souris est à gauche du centre
14 |   $H=$Souris_H-$Obj_G+$Decalage `déplacer l'objet à droite
15 | Sinon `la souris est à droite du centre
16 |   $H=-($Obj_D-$Souris_H+$Decalage) `déplacer l'objet à gauche
17 | Fin de si
18 |
19 |
20 | Si ($Souris_V<$Centre_V) `si la souris est au dessus du centre
21 |   $B=$Souris_V-$Obj_H+$Decalage `décaler vers le bas
22 | Sinon `la souris est en dessous du centre
23 |   $B=-($Obj_B-$Souris_V+$Decalage) `décaler vers le haut
24 | Fin de si
25 |
26 | `Déplacer le bouton en fonction des informations définies ci-dessus
27 | DEPLACER OBJET(bSpecial,$H,$B)
28 |

```

Vous pouvez copier le code dans la méthode `X_BOUTON_INCLIQUABLE`

Il ne vous reste plus qu'à visualiser le formulaire en utilisation, afficher la page des préférences et essayer de cliquer sur le bouton.

Alors ? Vous y arrivez ? (Si le bouton sort de l'écran, refermez le formulaire et recommencez, le bouton reprendra sa place initiale).

Cet exercice nous prouve 3 choses :

1°) Je ne suis pas très sérieux.

2°) On peut bien s'amuser tout en découvrant 4D et préparer les farces du 1er avril.

3°) La méthode n'est exécutée que lorsque l'événement coché survient.

Seul ce 3^e élément nous intéresse réellement dans le cadre de cet exercice. Restons sérieux ! :=)

Détaillons maintenant les principaux événements formulaires que vous allez utiliser à court terme en donnant un exemple d'utilisation pour chacun. A l'évidence, tous les objets ne disposent pas des mêmes événements ; il n'est pas possible par exemple de cocher l'événement "Sur données modifiées" pour un objet bouton, car on ne peut pas "saisir" son contenu (son titre).

Événement	Déclenchement	Commentaires
Sur début survol	Lorsque la souris "entre dans l'espace aérien" d'un objet	Nous l'avons vu dans l'exemple précédent. Les 2 autres événements liés : sur survol et sur fin survol.
Sur données modifiées	A la sortie d'un champ ou un variable dont le contenu a été modifié	On l'utilise principalement pour des contrôles de saisie, formatage (majuscules, minuscules), recherches, recalculs
Sur clic	Lors du clic sur un objet (principalement les boutons, popup, menus, ...)	Peut également être utilisé sur des objets saisissables

Evénements

Sur gain focus	L'objet vient de recevoir le focus (on a cliqué dessus, on est arrivé avec la touche tabulation ou on y est allé par programmation)	Cet événement est exécuté avant l'événement sur clic mais après l'événement sur début survol. Pour un bouton, les événements sont exécutés dans l'ordre : - sur début survol - sur survol - sur gain focus - sur clic - sur fin survol - sur perte focus Un champ ou une variable sur laquelle on clique : - sur début survol - sur survol - sur gain focus - sur données modifiées - sur perte focus - sur fin survol
Sur chargement	Juste avant qu'un formulaire soit affiché à l'écran ou utilisé lors d'une impression	C'est lors de cet événement qu'on effectue en général les initialisations.
Sur impression corps	Lors de l'impression d'un enregistrement	Permet par exemple de valoriser des variables de cumul, concaténation...

Cette liste vous permet de comprendre cette notion d'événement. Je vous renvoie à la documentation de 4D pour la description complète des événements.

Une précision importante que se posent toujours les débutants : "Dois-je cocher les événements du formulaire ou les événements des objets ?" La réponse est simple : Tout dépend de l'endroit où vous allez écrire votre méthode.

Certes me direz-vous, mais pourquoi écrire plutôt dans la méthode formulaire que dans la méthode objet ou inversement ?

Deuxième bonne question. Bravo vous suivez !

La méthode formulaire ne devrait en théorie contenir que ce qui concerne le traitement du formulaire en globalité (redimensionnement, appel extérieur, affichage ou masquage d'objets...). Les méthodes spécifiques à un objet (bouton, champ...) se trouvant en revanche sur l'objet.

Il arrive qu'on déplace certains traitements sur la méthode formulaire. Par exemple vous devez recalculer une valeur en fonction de 10 paramètres saisissables. A chaque paramètre modifié, il est nécessaire de refaire le calcul. Plutôt que de mettre la formule (ou appel de méthode) dans chacun des 10 champs paramètres, il est possible de déplacer ce calcul dans la méthode formulaire. Dans ce cas, c'est plus simple et centralisé, par contre, il est fort probable que le recalcul sera effectué plus souvent que nécessaire, notamment lors de la modification d'une zone qui n'entre pas dans le calcul final.

Emplacement	Avantages	Inconvénients
Méthode formulaire	Centralisé, traitement global, facile à maintenir	Risque d'être exécuté plus que nécessaire
Méthode objet	Spécifique, adapté, exécuté seulement quand c'est nécessaire. Permet facilement le portage d'un objet par simple Copier-Coller (surtout si vous utilisez les pointeurs)	Oblige à dupliquer la méthode ou l'appel de méthode dans chacun des objets.



ASTUCE

Pour gagner du temps dans l'écriture de votre code et le fiabiliser, pensez à paramétrer les macros (fichier "macros.xml"). Vous pourrez par exemple en créer une qui écrit à votre place le code suivant lorsque vous tapez "\$evt" dans votre méthode :

`$evt:=Evenement formulaire`

Au cas ou


`: ($evt=Sur données modifiées)`

`: ($evt=Sur chargement)`

Fin de cas

EXERCICE



Organisez la page Interventions comme indiqué sur la copie écran ci-dessous. Vous remarquerez en bas à droite une règle allant de 0 à 100 (tracée avec l'objet ) . Elle a pour fonction d'indiquer visuellement

et quantitativement l'état d'avancement de l'intervention de 0% à 100% :

The screenshot shows a software window with a toolbar at the top containing icons for 'Valider', 'Annuler', 'Première', 'Précédente', 'Suivante', 'Dernière', 'Supprimer', 'Imprimer', and 'Chercher'. Below the toolbar, there are several input fields:

- ID : [INTERV]
- ID_Technicien : [INTERV] [INTERVENTIONS]ID_Technicien [INTERVENTIONS]Objet
- ID_Matériel : [INTERVENTIONS]ID_M: Type_Intervention [INTERVENTIONS]Type_Interve
- Objet : [INTERVENTIONS]Objet
- Descriptif : [INTERVENTIONS]Descriptif
- Date_Début : [INTERVENTION] Heure_Début : [INTERVE]
- Date_Fin : [INTERVENTION] Heure_Fin : [INTERVE]
- Commentaires : [INTERVENTIONS]Commentaires

At the bottom right, there is a progress bar labeled 'Avancement' with a red indicator at 100% and a scale from 0 to 100.

Paramétrez la règle selon le modèle suivant :

Objets	
Type	Thermomètre
Nom de l'objet	TH_Avancement
Nom de la variable	TH_Avancement
Graduation	
Minimum	0
Maximum	100
Unité	10
Pas	5
Emplacement du libellé	Bas
Libellés	<input checked="" type="checkbox"/>
Graduation	<input checked="" type="checkbox"/>

Ensuite et c'est votre mission, il faut que la valeur de cette règle corresponde à la valeur du champ lorsqu'on ouvre un enregistrement Intervention. Puis, si on met à jour la règle en glissant à gauche ou à droite, vous devez mettre à jour le champ avec la valeur de la règle.



Voir corrigé page 252.

COMMENTAIRES



Je vous conseille au départ de mettre vos méthodes dans vos objets, vous aurez plus de souplesse pour corriger. Ensuite, commencez par utiliser des appels de méthodes génériques et enfin quand tout fonctionne correctement, voyez ce que vous pouvez déplacer de vos méthodes objets vers votre méthode formulaire.

Pour aller plus loin

Analysez bien le fonctionnement des événements car ils vous permettront de comprendre précisément à quel moment s'exécute votre programmation. Vous y trouverez aussi de nombreuses idées pour des mises en oeuvre performantes en utilisant par exemple les événements : sur glisser, sur déposer, sur nouvelle sélection, les trois 'sur survol' et bien évidemment les grands classiques sur clic, sur double clic, sur clic long, etc.

26

Tableaux, pop up, listbox

Objectif(s) pédagogique(s)	Utilisation des tableaux et association à des objets de formulaires
Durée estimée	30 minutes
Ce que nous allons utiliser	Programmation, variables et objets d'interface

Les tableaux font partie des incontournables de 4D. Pratiques, sans limite, dynamiques, à 1 ou 2 dimensions, ils sont un espace en mémoire qui peut s'afficher sur les formulaires par l'intermédiaire d'objets (popup, combo box, listbox, zones de défilement...).

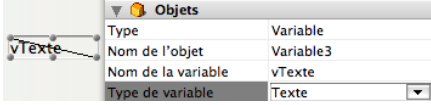
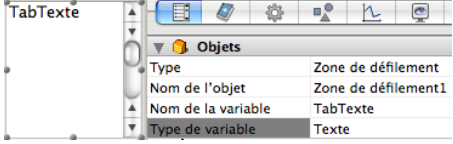
MISE EN OEUVRE

Nous avons déjà abordé le concept et la mise en œuvre de variables simples à une seule valeur. Un tableau est une variable multi-valuée dont on pourra lire ou écrire chacun des éléments. Comme une variable simple, un tableau doit être déclaré puis valorisé et enfin utilisé.

On définit pour un tableau le nombre de lignes qu'il contient et son type. Voici à titre de comparaison les différentes étapes de la vie d'une variable et d'un tableau :

Etape	Variable simple	Variable tableau
Initialisation	C_Texte(vTexte)	TABLEAU TEXTE(TabTexte;10) `10 lignes
Valorisation	vTexte:="Tascher de la Pagerie"	TabTexte{1}:="De Beauharnais" `ligne 1 TabTexte{2}:="Barras" `ligne 2 TabTexte{3}:="Bonaparte" `ligne 3 ...

Tableaux, pop up, listbox

Utilisation	<code>\$NbCar:=Longueur(vTexte)</code>	<code>\$Amant:=TabTexte{1}</code>
Effacer le contenu	<code>EFFACER VARIABLE(vTexte)</code> (le comportement est différent entre une application interprétée et compilée, voir la documentation 4D)	<code>TABLEAU TEXTE(TabTexte;0)</code>
Visualisation sur un formulaire	Donner le nom de la variable à un objet de type variable 	Donner le nom de la variable à un objet de type zone de défilement, pop up menu... 
Types	Entier, Entier long, Numérique, Alpha, Texte, Booléen, Date, Heure, Image, Blob, pointeurs	Identiques aux types de variables sauf Heure et Blob


On remarque que les similitudes sont nombreuses.

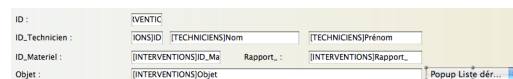
Vous connaissez presque tout des tableaux, il ne reste plus qu'à essayer en créant un premier tableau dans le formulaire DETAIL de la table INTERVENTIONS.

Nous devons donc :

- 1°) Créer le tableau en mémoire (tableau des "objets" d'intervention)
- 2°) Mettre un objet sur le formulaire capable de représenter le tableau en mémoire
- 3°) Vérifier que le tableau contient les bonnes informations
- 4°) Sélectionner une valeur et transférer cette valeur dans le champ *OBJET*

Pour simplifier les premières manipulations, nous allons mettre l'ensemble de la programmation sur l'objet défini au 2°). Ca nous permettra de réviser la notion d'événement.

- 1 Sélectionnez l'objet popup/liste déroulante  puis tracez-le à droite du champ OBJET.**



- 2 Nommez-le TabObjetInterventions**

Dans la méthode du popup, nous prévoyons deux événements (vérifiez qu'ils soient bien cochés) :

- Sur chargement : juste avant que le formulaire apparaisse à l'écran, nous allons initialiser le tableau et le valoriser
- Sur clic : lorsque l'utilisateur choisira une valeur, nous transférerons son choix dans le champ Objet.

Voici la méthode à écrire :

```

1 $evt:=Evenement formulaire
2 Au cas ou
3   ($evt=Sur chargement) `avant l'affichage du formulaire
4   TABLEAU TEXTE(TabObjetInterventions;3) `Définition du tableau (Type, Nom et nombre de lignes)
5   `remplissage des lignes du tableau
6   TabObjetInterventions(1):="Installation"
7   TabObjetInterventions(2):="Formation"
8   TabObjetInterventions(3):="Dépannage"
9
10  ($evt=Sur clic) `quand l'utilisateur choisit une ligne dans le tableau
11  Si (TabObjetInterventions#0) `si l'utilisateur a choisi une ligne
12  $LigneChoisie:=TabObjetInterventions `on mémorise la ligne choisie
13  $ValeurChoisie:=TabObjetInterventions($LigneChoisie) `on mémorise la valeur contenue dans cette ligne
14  [INTERVENTIONS]Objet:=$ValeurChoisie `on affecte la valeur au champ
15  Fin de si
16
17 Fin de cas
18

```

Vous avez certainement remarqué que le nom du tableau est utilisé parfois avec les accolades {}, parfois seul (lignes 11 et 12). Il s'agit dans ce dernier cas d'une variable (entier long) **créée automatiquement par 4D**. Cette variable, associée au tableau, sert d'indice (numéro de ligne) de tableau. C'est par son intermédiaire qu'on peut savoir quelle est la ligne choisie par l'utilisateur, ou forcer la sélection de telle ou telle ligne dans le popup menu.

C'est la raison pour laquelle vous verrez souvent écrit dans les bases exemples ou les développements des autres développeurs cette syntaxe plus concise :

```
[INTERVENTIONS]Objet:=TabObjetInterventions{TabObjetInterventions}
```

qu'on peut décrypter comme ceci :

"Objet := contenu du tableau {à la ligne choisie}"

Vous trouverez également, plus concise et beaucoup plus générique, cette syntaxe qui utilise la commande **Self** (pointeur vers l'objet dont la méthode est en cours d'exécution) :

```
[INTERVENTIONS]Objet:=Self->{Self->}
```

Quelle que soit la syntaxe utilisée, le fonctionnement est identique. Vous devez donc obtenir en saisie :

Form fields: ID: 0, ID_Technicien: [], ID_Matériel: [], Type_Intervention: [], Objet: []. Dropdown menu: Installation, Formation, Dépannage.

et après avoir choisi une valeur, cette valeur est transférée dans le champ Objet :

Form fields: ID: 0, ID_Technicien: [], ID_Matériel: [], Type_Intervention: [], Objet: Formation. Dropdown menu: Formation.

Voici un autre exemple avec un nouveau tableau qui permet de saisir l'heure de début d'intervention en présentant une amplitude horaire de 12 heures (8:00 à 20:00) par tranche de 30 minutes.

Pour la création du popup liste déroulante, procédez à l'identique puis créez la méthode :

```

1  $evt=Evenement formulaire
2  Au cas ou
3  [ $evt=Sur chargement ]
4      C_HEURE($HeureDeb) `variable contenant l'heure de départ
5      C_ENTIER LONG($TailleTableauHeures) `nombre de tranches horaires
6      $TailleTableauHeures=25 `on définit 12 heures par 1/2 heure + Borne basse
7      TABLEAU ENTIER LONG(TabHeuresDebut,$TailleTableauHeures) `initialisation du tableau de n lignes
8      $Heuredeb=?05:30:00? `heure de début
9
10     [ Boucle ($i;1;$TailleTableauHeures) `pour chaque ligne du tableau
11         TabHeuresDebut{$i}=$HeureDeb+(?00:30:00?*$i) `ajouter n tranches de 30 minutes à l'heure de départ
12     ]
13     Fin de boucle
14     [ $evt=Sur clic ] `lors de la sélection d'une ligne
15     [ Si (Self->#0) `si ue ligne est sélectionnée
16         `ci-dessous, on convertit la valeur Entier long en heure
17         [INTERVENTIONS]Heure_Début=Heure(Self->(Self->)) `attribuer la valeur de la ligne au champ Heure_Début
18     ]
19     Fin de si
20 ]
21 Fin de cas

```

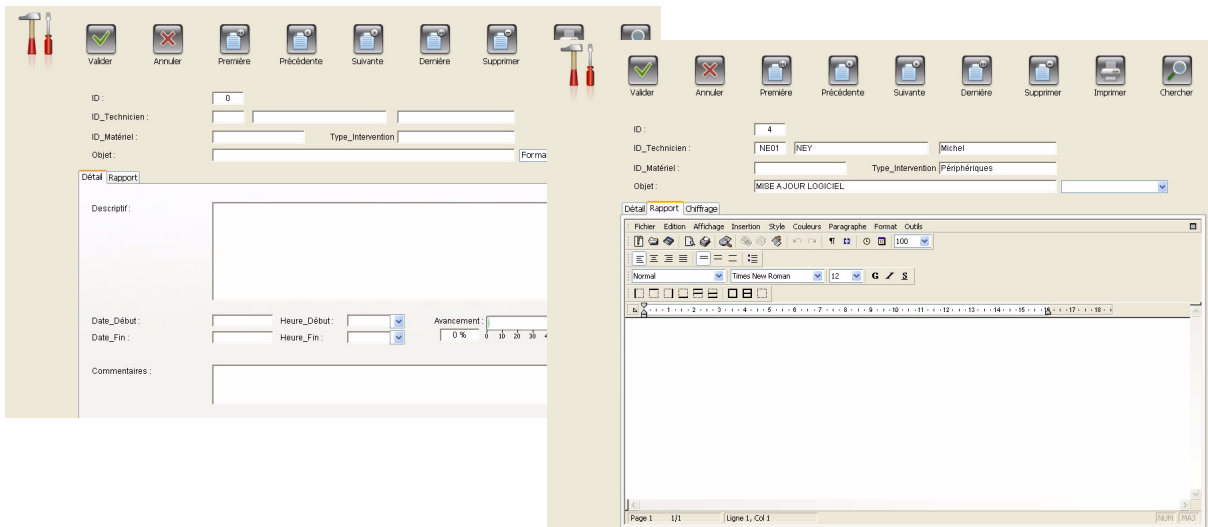
Vous constaterez dans la base suivante la manière dont j'ai mis en place ces tableaux d'heures. Un tableau d'heures (interprocess) est initialisé au démarrage de la base de données (méthode *INIT_Tableaux* appelée par la méthode *Initialisations*). Pour l'utiliser dans le contexte

des interventions, on copie le tableau au chargement de la méthode objet, ce qui optimise la programmation en évitant de recalculer le tableau à chaque fois, notamment parce que nous l'utilisons à plusieurs endroits (heure début, heure fin, etc.)

Troisième et dernier exemple : la mise en place d'un onglet.

Un onglet est un objet unique avec plusieurs titres (valeurs). Dans 4D, c'est un des objets capables de représenter un tableau.

Nous allons réorganiser le formulaire DETAIL de la table Intervention, en déplaçant en page 0 les informations devant figurer sur toutes les pages, en page 1 les informations de détail et en 2^e page le rapport d'intervention. Voici l'aperçu des pages une fois réorganisées :

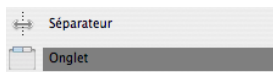


Vous remarquez que la partie supérieure est identique. Nous la placerons donc en page 0. L'onglet nous sert à la navigation, quelle que soit la page affichée, il est donc nécessaire de la placer en page 0. Nous verrons ensuite le contenu de la 2^e page.

Mettons en œuvre cette présentation :

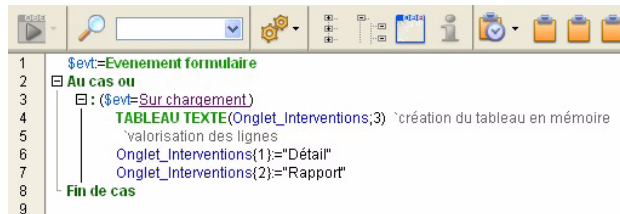
- 1 Affichez le formulaire en mode Développement.
- 2 Coupez de la page 1 les informations (7 champs + texte et Popup Objet)
- 3 Allez en page 0 et collez ce que vous venez de couper.

4 Sélectionnez l'objet Onglet et tracez-le dans le formulaire.



5 Nommez-le Onglet_Interventions (nom de variable).

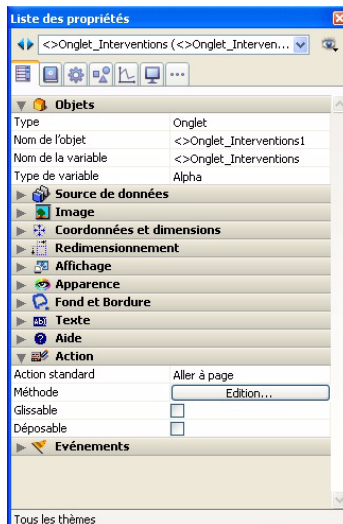
6 Editez sa méthode et recopiez ceci :



7 Créez la deuxième page de votre formulaire (laissez-la vide pour l'instant, nous la remplirons ensuite).

Vous pouvez tester la navigation et le passage de page en page avec votre onglet. Cela fonctionne et pourtant nous n'avons écrit nulle part le comportement que doit avoir l'onglet lorsqu'on clique dessus... vous vous étiez très justement posé la question :=)

Non, Harry Potter n'a pas exercé ses pouvoirs sur notre développement :-). C'est tout simplement 4D qui attribue par défaut à un onglet l'action standard "Aller à page" :



Nous pourrions bien sûr désactiver cette action standard et gérer par programmation le comportement de l'onglet.

EXERCICE



Créez un popup/liste déroulante pour indiquer l'heure de fin de l'intervention.



Voir corrigé page 253.

COMMENTAIRES



Vous l'avez constaté, les tableaux deviennent vite très utiles voire indispensables.

Dans 4D, un tableau ne contient que des éléments de même type. Vous ne pouvez pas avoir un tableau avec une ligne Alpha, une autre Date et une troisième Heure. Dans ce cas, vous pouvez utiliser un tableau de pointeurs qui pointera potentiellement vers des variables de types différents.

Pour aller plus loin

Comme évoqué à la leçon sur les pointeurs, vous pouvez combiner les pointeurs et les tableaux pour obtenir les "tableaux de pointeurs".

Pensez également qu'une Listbox est une série de tableaux accolés (de même dimension N).

27

4D Write

Objectif(s) pédagogique(s)	Installation, paramétrage et utilisation du plug-in 4D Write
Durée estimée	15 minutes
Ce que nous allons utiliser	Dossier des plug-ins, éditeur de formulaires

4D permet très facilement l'ajout de plug-ins. Ils sont fournis par 4D SAS, par d'autres éditeurs ou bien par vous-même.

Dans cette leçon, nous aborderons les phases d'installation et de paramétrage du plug-in 4D Write, permettant de disposer de fonctions de traitement de texte au sein de l'application 4D.

MISE EN OEUVRE

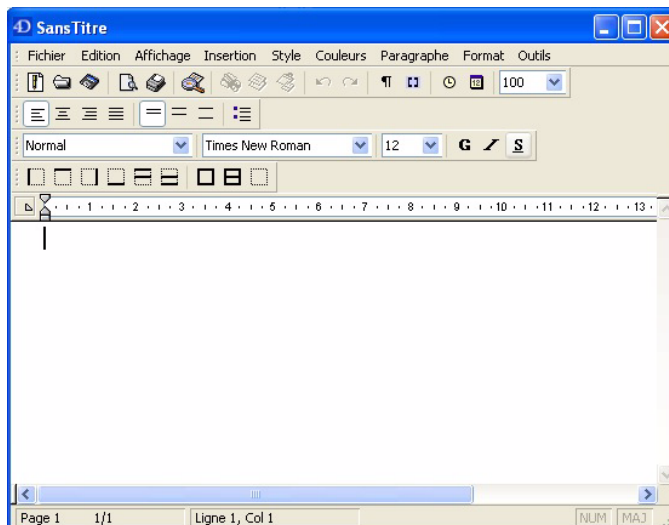
Commençons par installer le plug-in :

- 1 Quittez votre application.
- 2 Créez un dossier nommé "Plug-ins" au même niveau que votre fichier de structure (sur Mac OS, placez ce dossier dans le paquet Content:plug-ins).
- 3 Glissez le plug-in 4D Write dans ce dossier Plug-ins.
- 4 Relancez votre application (si vous ne disposez pas de licence, un message vous indique que la durée d'utilisation autorisée est de 30 minutes).

Votre plug-in est maintenant installé. Pour vérifier qu'il soit bien disponible vous pouvez :

- Choisir "Afficher la table courante" dans le menu "Enregistrements"
- Sélectionner **4D Write** dans le menu **Outils**.


Vous obtenez la fenêtre suivante :



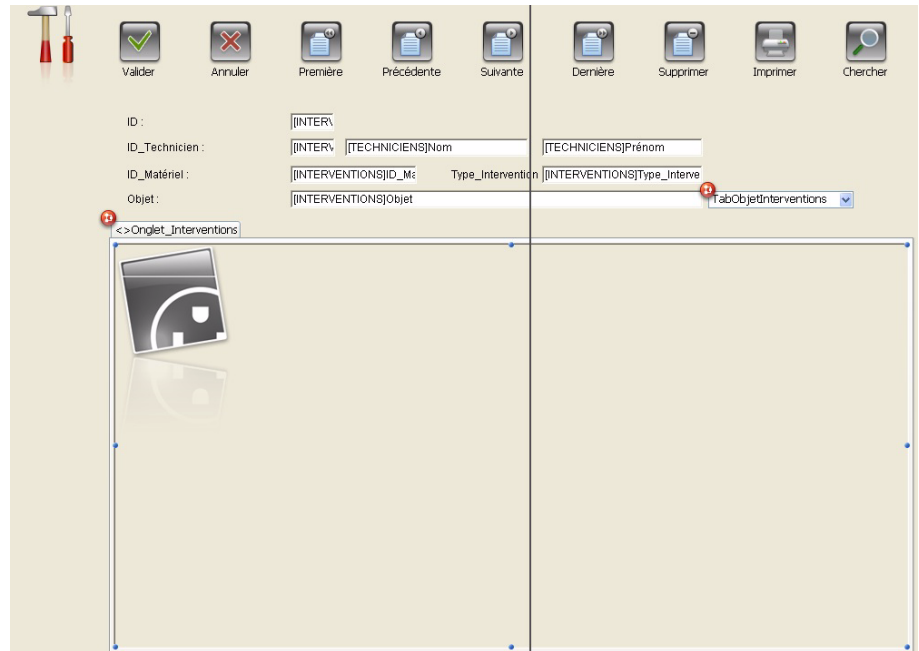
Votre plug-in est bien actif, vous pouvez refermer la fenêtre. Pour plus d'informations sur l'utilisation, les fonctionnalités et la programmation de 4D Write, je vous renvoie à sa documentation.

Je vous avais prévenu, rien de plus simple que d'installer un plug-in :=) Maintenant, nous allons associer une zone 4D Write à un champ, ce qui nous permettra de disposer d'un traitement de texte directement dans notre formulaire de saisie.

A l'exercice précédent, nous avons ajouté une page au formulaire DETAIL de la table Interventions. C'est dans cette deuxième page que nous allons installer le plug-in :

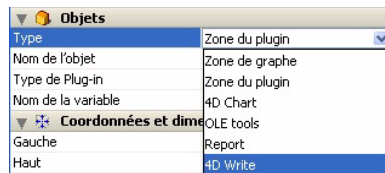
- 1 Ouvrez le formulaire DETAIL.
- 2 Allez en page 2.
- 3 Cliquez sur l'outil Zone de plug-in  .
- 4 Tracez une zone dans la partie inférieure de votre formulaire (dans la zone de l'onglet).

Votre formulaire ressemble à ceci :

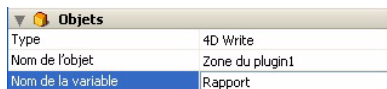


Nous devons maintenant paramétrer cette zone (en fait c'est une variable) afin qu'elle contienne le plug-in 4D Write :

5 Dans la propriété "Type" déroulez la liste et choisissez 4D Write :



6 Tapez "Rapport" comme nom de la variable :

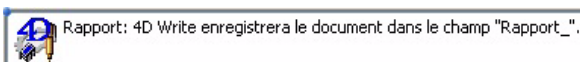


A ce stade nous disposons d'une zone de traitement de texte utilisable dans le formulaire. Si nous regardons le message (en rouge) affiché en haut de la zone, nous comprenons que pour associer (automatiquement) le contenu de cette zone à un champ de la table,

nous devrions avoir un champ dont le nom est "Rapport_" (le _ final est important) :

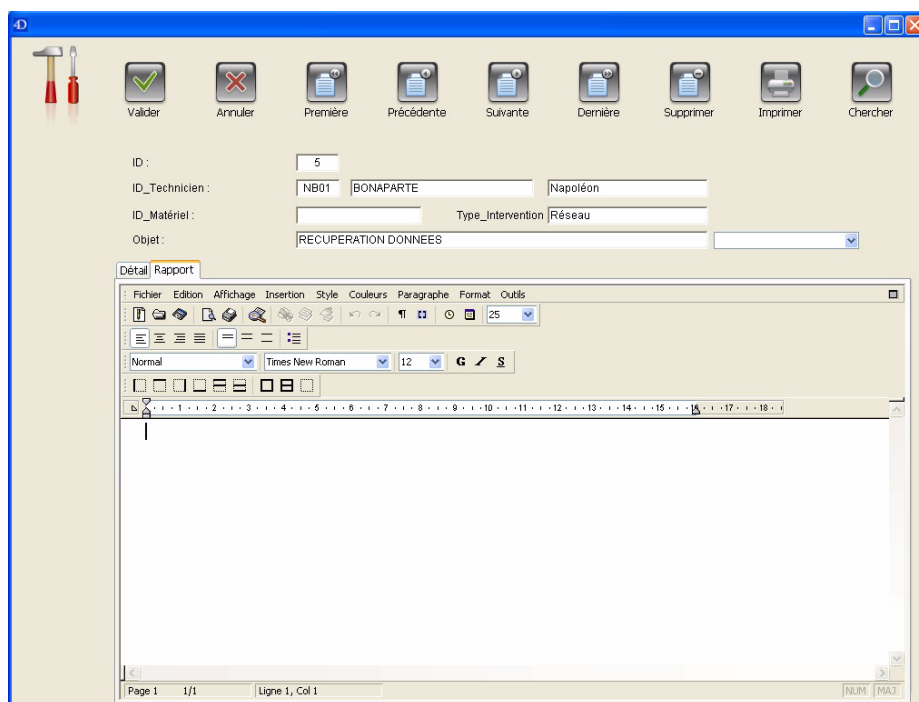


7 Créez le champ **Rapport_** de type **BLOB** dans la table **INTERVENTIONS**. Automatiquement, au retour dans le formulaire, le message indique :



Voilà, c'est terminé ! Maintenant, à chaque fois que nous allons créer ou modifier un rapport dans un enregistrement INTERVENTION, il sera automatiquement sauvegardé par 4D dans le champ **Rapport_** et restitué à l'ouverture de l'enregistrement, et tout ça, sans aucune programmation.

Vous pouvez tester en ajoutant et modifiant des enregistrements Interventions en mode utilisateur :



EXERCICE



Réalisez la même installation avec le plug-in 4D View (tableur) en créant une troisième page dans le formulaire DETAIL de la table Interventions.

Dans l'onglet, nommez cette 3e page CHIFFRAGE.



Voir corrigé page 253.

COMMENTAIRES



Certains plug-ins sont intégrés à l'offre 4D (4D Internet Commands, 4D Pack...) je vous conseille de les installer d'emblée dans vos développements. Aujourd'hui il est rare de développer une application sans prendre en compte des flux FTP, des envois ou réception de mails, ou le démarrage programmé d'autres applications.

Il vous sera également possible de développer votre propres plug-ins avec le "Plug-in SDK". L'outil vous permet de définir vos points d'entrée, les variables en entrée et en sortie ainsi que de nombreux paramètres liés à la génération du code. Ensuite, vous pouvez générer votre projet. Le code d'appel des bibliothèques est généré automatiquement. Il ne vous reste qu'à écrire votre code spécifique.

Les plug-ins apportent une souplesse et une puissance de fonctionnement très importante. Je vous conseille de regarder la liste des plug-ins fournis par 4D SAS ainsi que ceux des éditeurs tierces parties. Vous gagnerez dans la plupart des cas un temps considérable en vous appuyant sur le travail déjà réalisé et maintenu par ceux qui ont été confrontés aux mêmes problématiques que vous.

Pour aller plus loin

Voir le fonctionnement ou les options proposées par 4D Chart, 4D Internet Commands, 4D Pack...

28

Fenêtre et navigation

Objectif(s) pédagogique(s)	<ul style="list-style-type: none"> • Comprendre l'interaction possible entre la programmation et la gestion de l'interface. • Utiliser les propriétés de dimensionnement automatique des objets.
Durée estimée	30 minutes
Ce que nous allons utiliser	Editeur de formulaires, Palette de propriétés Editeur de méthodes (événements sur chargement)

La création de l'interface est très consommatrice de temps dans un développement (jusqu'à 70% de la charge selon les développements). Il est donc important d'en connaître les principaux rouages afin de ne pas réinventer ce que 4D vous permet de faire facilement (redimensionnement, agrandissement d'objets, gestion de la case de fermeture...).

MISE EN OEUVRE

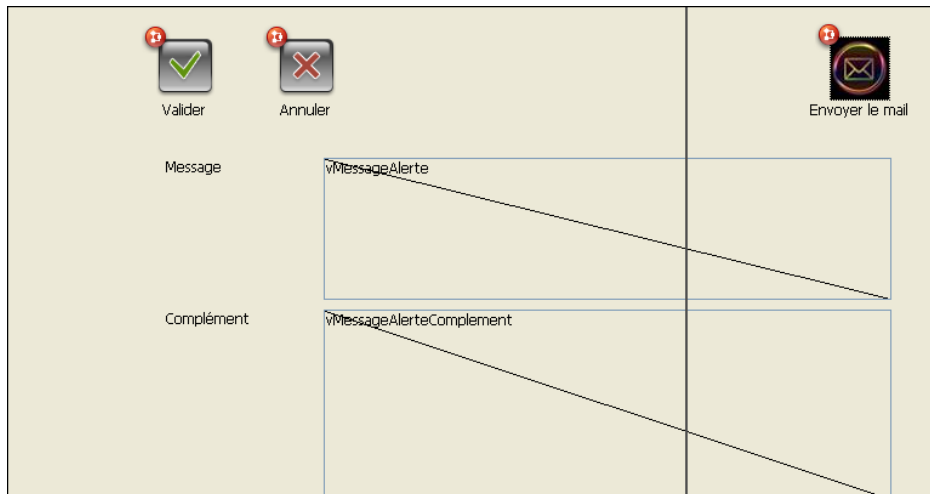
Dans notre formulaire DETAIL de la table Interventions, nous voulons disposer d'un bouton d'envoi de mail. Ce mail sera adressé au technicien en charge de l'intervention.

Pour cela, il faut que nous ajoutions ce bouton et l'interface de saisie du mail. Ensuite, il faudra envoyer le mail (ce sera l'objet de la leçon suivante).

1 Créez le formulaire de saisie du mail en utilisant un formulaire projet.

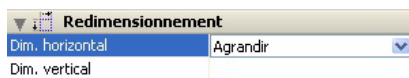
D'emblée, nous prévoyons d'utiliser ce formulaire dans d'autres circonstances. Toutes les zones du formulaire seront donc des variables

que nous pourrons alimenter avec le contenu des champs de telle ou telle table.



Profitons-en pour rendre les zones auto-ajustables. Leur largeur et/ou hauteur s'adaptera à la taille de la fenêtre en cas de redimensionnement.

2 Sélectionnez toutes les variables puis modifiez la propriété "Dim. Horizontal".



3 Ensuite, seulement pour la variable *vContenu*, mettez également en "Agrandir" la propriété "Dim.Vertical" :



Avec ces propriétés Dim., vous commencez à voir les dessous affriolants de 4D dont les objets s'adaptent à vos envies, en collant à la taille du formulaire.

Nous devons maintenant indiquer le type de ces variables.

4 Placez cette ligne de programme dans la méthode formulaire pour l'événement sur chargement.

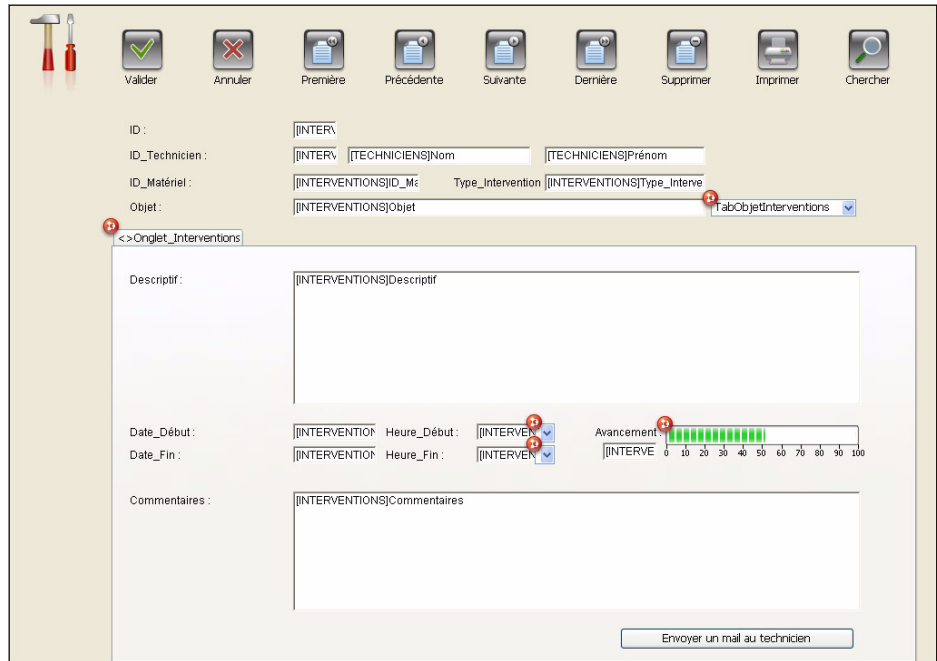
C'est en effet l'emplacement idéal pour ce genre de traitement.



5 Ajoutez un bouton pour envoyer le mail (nous le programmerons lors de la prochaine leçon).

Vous pouvez l'ajouter à la bibliothèque d'images. Il s'agit du fichier "Mail" dans le dossier "Images_png". Glissez ensuite ce bouton de la bibliothèque vers votre formulaire.

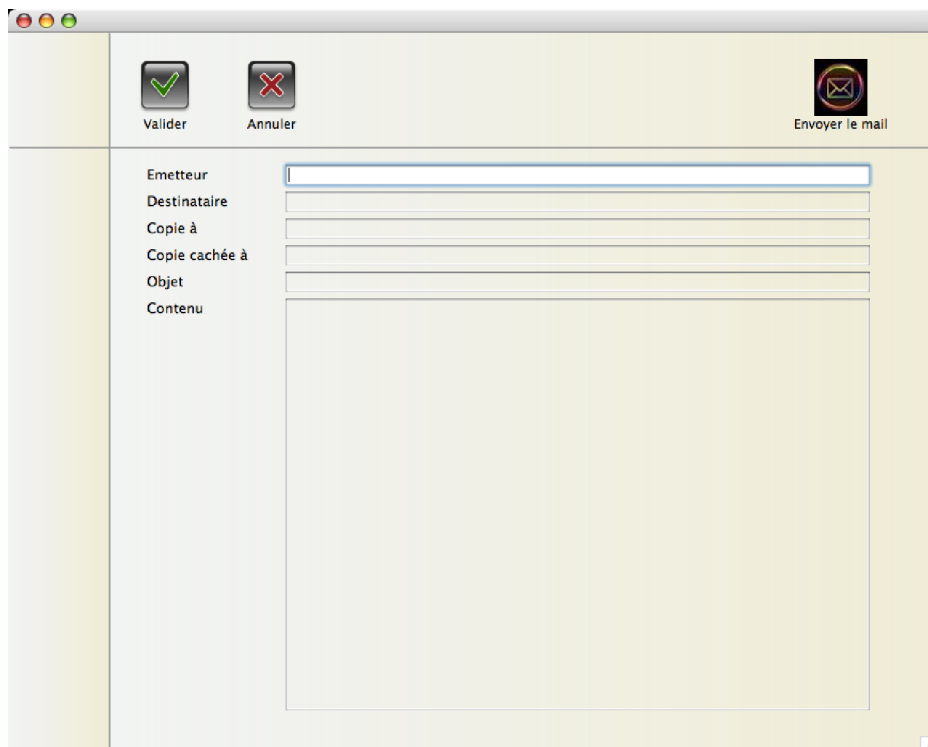
Le dialogue est prêt. Nous pouvons créer, en bas du formulaire DETAIL Intervention, le bouton d'appel de ce dialogue :



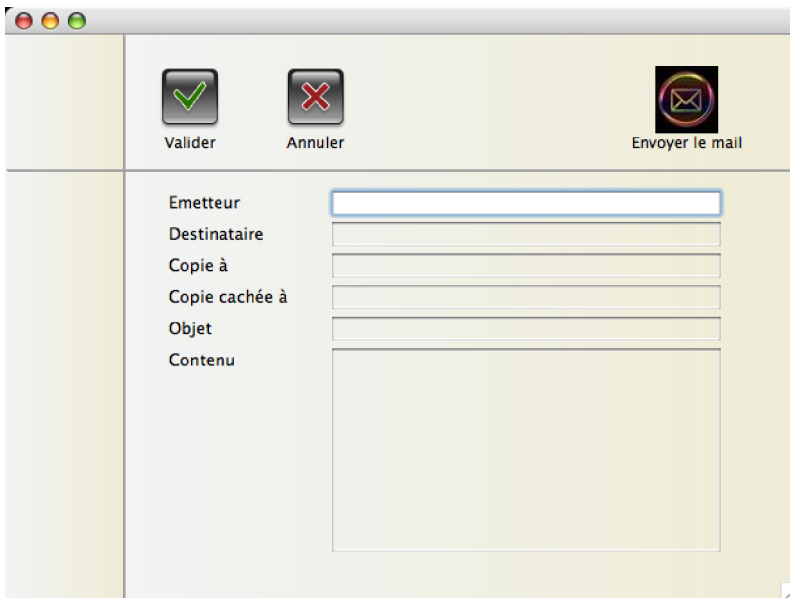
La méthode du bouton permettant d'afficher le dialogue est de la forme :



Testons maintenant le fonctionnement de ce dialogue. Vous devez obtenir cet écran :



Si vous redimensionnez la fenêtre, tous les champs doivent s'agrandir et le bouton d'envoi de mail doit se déplacer ainsi que le texte associé :



Il nous reste à transférer les informations utiles des champs de la table Intervention vers les variables de ce formulaire (email destinataire, objet et contenu). Voici une manière d'adapter le bouton d'appel du dialogue de mail :

```

1 `valoriser les variables pour faciliter la saisie
2 vEmetteur="prof@4d.fr"
3 vDestinataire:[TECHNICIENS]Email
4 vDestinataireCopie=""
5 vDestinataireCopieCachee=""
6 vObjet="Intervention prévue le "+Chaine([INTERVENTIONS]Date_Début)+" à "+Chaine([INTERVENTIONS]Heure_Début)
7 vContenu=[INTERVENTIONS]Objet+(Caractere(13)*2)+[INTERVENTIONS]Descriptif
8
9
10 $Fenetre:=Créer fenetre formulaire("MAIL";Fenêtre standard ;Centrée horizontalement;Centrée verticalement)
11 `afficher le dialogue
12 DIALOGUE("MAIL")
13 `une fois le dialogue refermé, refermer la fenetre
14 FERMER FENETRE($Fenetre)
15

```

Lors de la saisie d'une intervention, on peut envoyer un mail au technicien avec les informations souhaitées :

Valider Annuler Envoyer le mail



Emetteur prof@4d.fr
Destinataire napoleon@4d.fr
Copie à
Copie cachée à
Objet Intervention prévue le 10/05/07 à 11:00:00
Contenu Mise en place d'une unité de sauvegarde
Le système de protection et de gestion de la sureté n'est plus assuré. Il faut doubler le système de sauvegarde et installer de nouvelles unités en Raid 5.

EXERCICE



4D dispose de la commande ALERTE pour afficher des messages. Vous constaterez rapidement qu'il est souvent nécessaire d'agrémenter les messages avec des informations complémentaires. En prévision de vos développements futurs, vous allez mettre en place un formulaire paramétrable destiné à remplacer la commande ALERTE et à afficher des messages plus personnalisés.

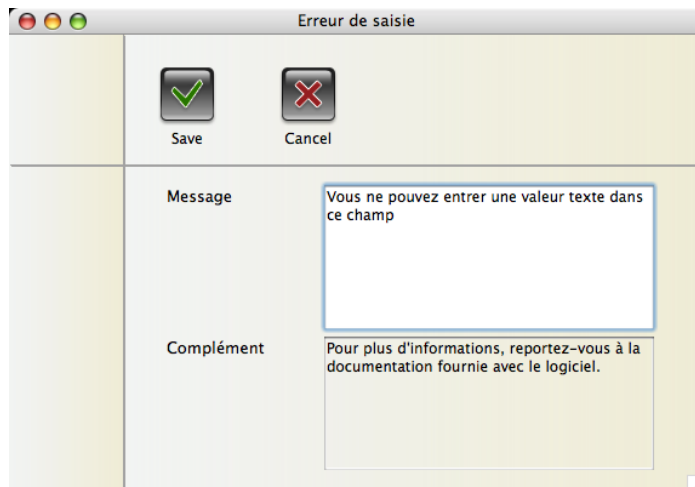
Voici le modèle de formulaire à créer :

	 Valider	 Annuler
Message	vMessageAlerte	
Complément	vMessageAlerteComplement	

Ensuite, ce dialogue est appelé par une méthode que vous devez créer. Cette méthode `ALERTE_DIALOGUE` reçoit 3 paramètres texte :

- le titre de la fenêtre,
- le message d'alerte,
- un complément éventuel.

Vous devez pouvoir afficher un message semblable à celui-ci en cas d'erreur :




Voir corrigé page 255.

COMMENTAIRES



Pensez également que ce dialogue d'alerte est un formulaire à part entière. Il peut donc contenir plusieurs pages, un onglet, des boutons, des tableaux et tout ce qui vous semblera nécessaire. Vous pourriez même proposer l'envoi automatique d'un mail au service assistance avec des informations sur l'utilisateur, la machine, la date, l'heure, le message d'erreur, etc. ou tout simplement l'enregistrement de ces informations dans un fichier de log accessible ensuite à l'assistance technique.

Vous pouvez également paramétrer le formulaire pour qu'il affiche ou masque tel ou tel bouton, qu'il ajoute un URL cliquable permettant l'accès à une aide en ligne, le choix de la langue des messages, paramétrer les messages à partir d'une table renseignée par l'utilisateur, etc. Une fois le principe compris, la seule limite sera votre imagination

Pour aller plus loin

Les types de fenêtres, gestion de la case de fermeture

29

4D Internet Commands

Objectif(s) pédagogique(s)	Deux manières d'envoyer un email : <ul style="list-style-type: none"> • la méthode simple, efficace, mais peu paramétrable • une méthode un peu moins simple, mais offrant de nombreuses possibilités, avec notamment l'authentification SMTP et l'envoi d'un fichier
Durée estimée	45 minutes
Ce que nous allons utiliser	Plug-in 4D Internet Commands (Commandes SMTP), Editeur de méthodes

Vous rêvez d'envoyer facilement des mails à partir de votre base de données. Nous avons mis en place l'interface nécessaire dans la leçon précédente. Le plug-in 4D Internet Commands propose deux méthodes pour envoyer des mails. L'une est simple, efficace, mais peu paramétrable et l'autre est plus souple mais un peu plus longue à écrire.

Nous n'envisageons pas ici de conserver les mails envoyés. Cependant, si vous souhaitez le faire, vous pouvez soit créer une table avec les champs nécessaires, soit stocker les informations dans un blob de la table à partir de laquelle le mail a été envoyé, soit tout simplement envoyer le mail en copie à une adresse particulière.

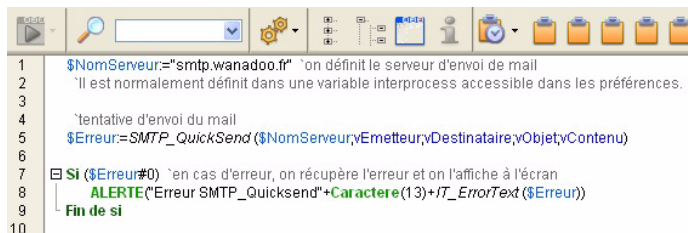
MISE EN OEUVRE

Pour commencer nous avons besoin des commandes du plug-in 4D Internet Commands.

Installez-le comme vous l'avez fait avec 4D Write et 4D View puis redémarrez votre base de données.

- 1 Ouvrez le formulaire projet MAIL puis affichez la méthode du bouton d'envoi de mail (en haut à droite).

La méthode la plus rapide est d'utiliser la commande SMTP_Quicksend de la manière suivante :



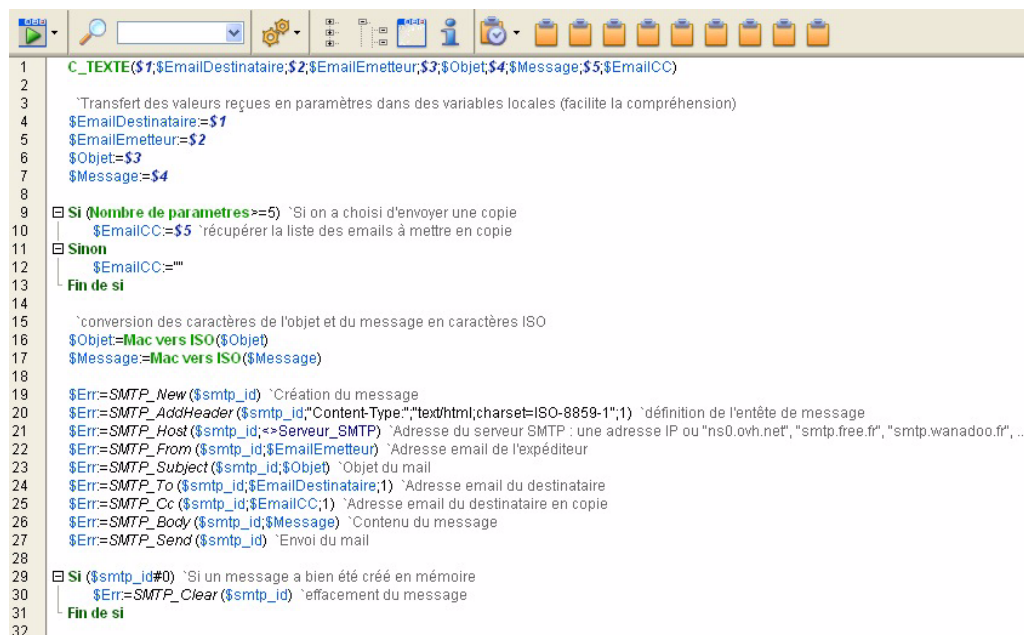
```

1  $NomServeur:="smtp.wanadoo.fr" `on définit le serveur d'envoi de mail
2  `Il est normalement défini dans une variable interprocess accessible dans les préférences.
3
4  `tentative d'envoi du mail
5  $Erreur:=SMTP_QuickSend($NomServeur;yEmetteur;yDestinaire;yObjet;yContenu)
6
7  Si ($Erreur#0) `en cas d'erreur, on récupère l'erreur et on l'affiche à l'écran
8  ALERTE("Erreur SMTP_Quicksend"+Caractere(13)+IT_ErrorText ($Erreur))
9  Fin de si
10

```

Vous remarquez que cette commande ne permet pas d'ajouter des copies, copies cachées et/ou des documents. Elle reste cependant très pratique et rapide à mettre en œuvre.

La deuxième solution proposée par le plug-in est d'utiliser une série de commandes unitaires qui définissent chacune un paramètre du mail. En voici une version simplifiée :



```

1  C_TEXTE($1;$EmailDestinaire;$2;$EmailEmetteur;$3;$Objet;$4;$Message;$5;$EmailCC)
2
3  `Transfert des valeurs reçues en paramètres dans des variables locales (facilite la compréhension)
4  $EmailDestinaire:=$1
5  $EmailEmetteur:=$2
6  $Objet:=$3
7  $Message:=$4
8
9  Si (Nombre de paramètres >= 5) `Si on a choisi d'envoyer une copie
10 $EmailCC:=$5 `récupérer la liste des emails à mettre en copie
11 Si non
12 $EmailCC:=""
13 Fin de si
14
15 `conversion des caractères de l'objet et du message en caractères ISO
16 $Objet:=Mac vers ISO($Objet)
17 $Message:=Mac vers ISO($Message)
18
19 $Err:=SMTP_New($smtp_id) `Création du message
20 $Err:=SMTP_AddHeader($smtp_id;"Content-Type","text/html;charset=ISO-8859-1";1) `définition de l'entête de message
21 $Err:=SMTP_Host($smtp_id;<=>Serveur_SMTP) `Adresse du serveur SMTP : une adresse IP ou "ns0.ovh.net", "smtp.free.fr", "smtp.wanadoo.fr", ...
22 $Err:=SMTP_From($smtp_id;$EmailEmetteur) `Adresse email de l'expéditeur
23 $Err:=SMTP_Subject($smtp_id;$Objet) `Objet du mail
24 $Err:=SMTP_To($smtp_id;$EmailDestinaire;1) `Adresse email du destinataire
25 $Err:=SMTP_Cc($smtp_id;$EmailCC;1) `Adresse email du destinataire en copie
26 $Err:=SMTP_Body($smtp_id;$Message) `Contenu du message
27 $Err:=SMTP_Send($smtp_id) `Envoi du mail
28
29 Si ($smtp_id#0) `Si un message a bien été créé en mémoire
30 $Err:=SMTP_Clear($smtp_id) `effacement du message
31 Fin de si
32

```

Cette version, bien que fonctionnelle, n'est pas pleinement satisfaisante car elle ne prend pas en compte tous les paramètres et ne tient pas compte des éventuels retours d'erreurs.

Voici une méthode plus complète, qui reste perfectible. Elle permet par exemple de n'ajouter qu'un seul fichier joint et ne prend pas en compte les copies cachées ou le reply-to :

```

1  C_TEXTE($1,$EmailDestinaire,$2,$EmailEmetteur,$3,$Objet,$4,$Message,$5,$EmailCC)
2  C_BOOLEAN($0,$6,$MessageAlerte)
3
4  `Transfert des valeurs reçues en paramètres dans des variables locales (facilite la compréhension)
5  $EmailDestinaire=$1
6  $EmailEmetteur=$2
7  $Objet=$3
8  $Message=$4
9
10  Si (Nombre de paramètres >= 5) `Si on a choisi d'envoyer une copie
11  | $EmailCC=$5 `récupérer la liste des emails à mettre en copie
12  Sinon
13  | $EmailCC=""
14  | Fin de si
15
16  Si (Nombre de paramètres >= 6) `permet d'afficher ou non les messages d'alerte
17  | $MessageAlerte=$6 `
18  Sinon
19  | $MessageAlerte=Vrai `par défaut on affiche les messages d'alerte
20  | Fin de si
21
22  $Objet=Mac vers ISO($Objet) `conversion des caractères de l'objet en caractères ISO
23  $Message=Mac vers ISO($Message) `conversion des caractères du message en caractères ISO
24
25  $Err=SMTP_New($smtp_id) `Création du message
26  Si ($Err=0)
27  | $Err=SMTP_AddHeader($smtp_id,"Content-Type","text/html;charset=ISO-8859-1";1) `définition de l'entête du message
28  | Fin de si
29  Si ($Err=0)
30  | $Err=SMTP_Host($smtp_id,<=>Serveur_SMTP) `une adresse IP ou "ns0.ovh.net", "smtp.free.fr" "smtp.wanadoo.fr" selon les cas `Adresse du serveur SMTP
31  | Fin de si
32  Si ($Err=0)
33  | $Err=SMTP_From($smtp_id,$EmailEmetteur) `Adresse email de l'expéditeur
34  | Fin de si
35  Si ($Err=0)
36  | $Err=SMTP_Subject($smtp_id,$Objet) `Objet du mail
37  | Fin de si
38  Si ($Err=0)
39  | $Err=SMTP_To($smtp_id,$EmailDestinaire;1) `Adresse email du destinataire
40  | Fin de si
41  Si ($Err=0)
42  | $Err=SMTP_Cc($smtp_id,$EmailCC;1) `Adresse email du destinataire en copie
43  | Fin de si
44  Si ($Err=0)
45  | $Err=SMTP_Body($smtp_id,$Message) `Contenu du message
46  | Fin de si
47  Si ($Err=0)
48  | $Err=SMTP_Attachment($smtp_id,vChemin;2;1) `Fichier joint
49  | Fin de si
50  Si ($Err=0)
51  | $User_SMTP="UserSMTP"
52  | $MDP_SMTP="mot de passe SMTP"
53  | $Err=SMTP_Auth($smtp_id,$User_SMTP,$MDP_SMTP) `authentification sur le serveur SMTP
54  | Fin de si
55  Si ($Err=0)
56  | $Err=SMTP_Send($smtp_id) `Envoi du mail
57  | Fin de si
58
59  $0=($Err=0) `si on n'a pas d'erreur, c'est que le mail est bien parti
60
61  Si ($Err#0)
62  | Si ($MessageAlerte) `si on veut afficher les alertes (cas par défaut)
63  | | ALERTE("erreur n°"+Chaine($Err)+<=>CR+IT_ErrorText($Err))
64  | | Fin de si
65  | Fin de si
66
67  Si ($smtp_id#0) `Si un message a bien été créé en mémoire
68  | $Err=SMTP_Clear($smtp_id) `effacement du message `si l'effacement ne s'est pas bien passé
69  | Fin de si

```

Mettez en place cette méthode d'envoi de mails (vous pouvez copier le texte dans la méthode Email_Envoi de la base corrigée).



Vous aurez remarqué l'utilisation de la variable $\langle \rangle$ Serveur_SMTP qui est paramétrée à l'initialisation de la base. Si vous ne valorisez pas cette variable, le plug-in 4D Internet Commands ne pourra pas envoyer vos mails et vous retournera un message d'erreur.

EXERCICE



Mettez en place une méthode qui permet d'envoyer un mail à une adresse mail définie lors de l'affichage des messages d'erreurs (exercice réalisé à la leçon précédente).



Voir corrigé page 256.

COMMENTAIRES



Avec 4D Internet Commands vous pouvez réaliser un logiciel de mail complet. Ce n'est pas l'objectif ici. L'intérêt est de savoir que toutes les fonctions nécessaires à la gestion d'internet (mail, ftp, smtp, http, tcp...) sont disponibles dans 4D.

L'envoi de mail est un très bon exercice pour commencer à programmer des échanges sur Internet. Vous pourrez le compléter en ajoutant dans vos messages des balises HTML qui permettent de mettre en gras, italique.... certaines parties de vos emails. Pensez dans ce cas à définir les en-têtes adéquats afin que votre mail HTML soit reconnu de la plupart des navigateurs.

Pour aller plus loin

Gestion des flux TCP, FTP avec 4D Internet commands

30 Mots de passe et groupes

Objectif(s) pédagogique(s)	Comment gérer les groupes et utilisateurs, leur attribuer des droits et garantir la pérennité lors des mises à jour.
Durée estimée	10 minutes
Ce que nous allons utiliser	Editeur de mots de passe, utilisateurs et groupes, Propriétés des méthodes

4D dispose d'un système de gestion des groupes, utilisateurs et permet de limiter les accès à différents objets ou actions :

- Tables : charger (voir), créer, modifier, supprimer
- Exécution des méthodes
- Affichage des formulaires
- Sélection de lignes de menus
- Accès aux plug-ins
- Accès au mode Développement

Il est également possible de gérer l'appartenance aux groupes, le contrôle des mots de passe, etc. par programmation.

MISE EN OEUVRE

Dans toute base de données 4D, il y a au moins deux utilisateurs

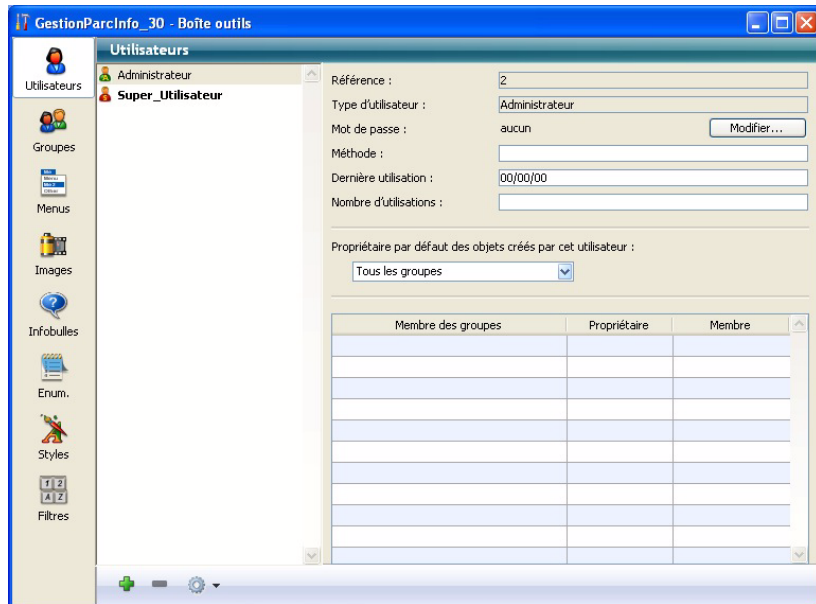
- le Super_Utilisateur qui a tous les droits sur le développement,
- l'Administrateur qui est en principe un utilisateur du client qui peut administrer la base de données et notamment réaliser les attributions de droits d'accès.

Tant que le Super_Utilisateur n'a pas de mot de passe, la base de données s'ouvre librement.

Pour afficher la liste des utilisateurs, affichez la boîte à outils :



La liste des utilisateurs est sélectionnée par défaut :





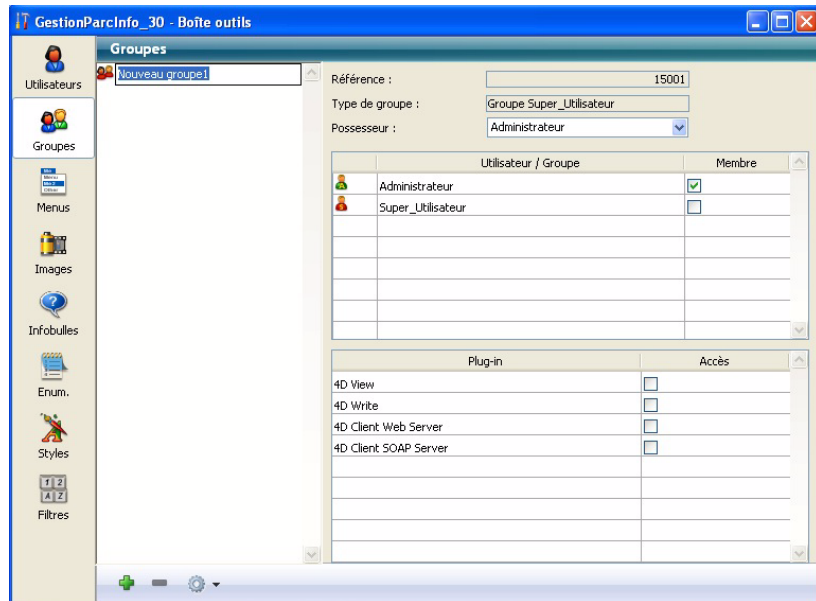
Il suffit de cliquer sur le Super_Utilisateur puis sur le bouton "Modifier" pour lui attribuer un mot de passe.

ATTENTION : Le mot de passe tient compte des minuscules et majuscules. Comme tout mot de passe, conservez-le précieusement car il vous sera impossible de le récupérer si vous l'oubliez.

Vous pouvez ensuite ajouter des utilisateurs. Attention, les utilisateurs créés par le Super_Utilisateur ne sont pas modifiables par l'Administrateur et inversement. C'est normal, car l'un est censé être le responsable des développeurs et l'autre l'administrateur côté client. Ils ne doivent donc pas interférer dans les droits.

Actuellement, puisqu'aucun mot de passe n'est défini, la base est démarrée en Super_Utilisateur.

Pour créer des groupes, cliquez sur  puis sur le bouton  .



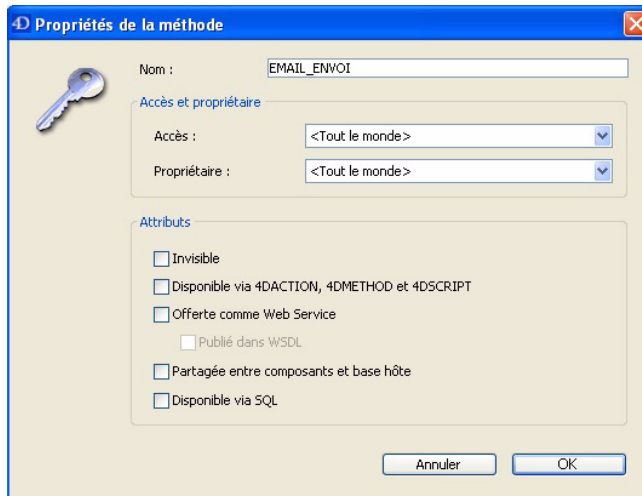
Saisissez le nom du groupe puis cochez les utilisateurs qui en font partie, ainsi que les éventuels droits d'accès aux plug-ins.

Un groupe peut être membre d'un autre groupe. Dans ce cas le groupe enfant hérite des droits du groupe parent.

Une fois vos groupes et utilisateurs créés et la répartition dans les groupes faite, vous pouvez indiquer l'utilisation que vous souhaitez faire de vos groupes.

Le principe est d'indiquer quel groupe a le droit de faire telle ou telle manipulation. Sur une méthode, vous pouvez indiquer que tel groupe

est propriétaire (a le droit de la modifier en développement) et que tel groupe d'utilisateurs a accès à la méthode et peut l'exécuter :



Si une personne non autorisée essaye d'exécuter la méthode, un message d'information lui explique que ses droits d'accès ne le lui permettent pas.

Par programmation, vous pouvez utiliser les commandes **Utilisateur courant** et **Appartient au groupe** pour intervenir sur les accès à telle ou telle partie de votre développement.

EXERCICE



Créez quelques utilisateurs (vous pouvez ne pas leur attribuer de mot de passe dans un premier temps durant vos tests). Lorsque vous

redémarrez votre base de données, vous devez obtenir un dialogue comme celui-ci :



Regardez dans la documentation la syntaxe des commandes **Utilisateur courant** et **Appartient au groupe** puis interdisez l'accès aux pages 7 et 8 du formulaire NAVIGATION à tout utilisateur qui ne fait pas partie du groupe "PARAMETRAGES". Il vous faut créer également ce groupe et lui attribuer des utilisateurs (dont l'Administrateur).



Voir corrigé page 257.

COMMENTAIRES



Les utilisateurs et groupes sont conservés dans la structure de 4D. Ce fonctionnement implique quelques précautions lorsque vous effectuez des mises à jour sur le site de production car toute modification faite par l'administrateur (attribution d'un utilisateur à un groupe, etc.) risque d'être écrasée par votre nouvelle structure (celle dans laquelle vous avez fait des modifications).

Mots de passe et groupes

Il est possible pour l'administrateur d'exporter puis de réimporter l'ensemble des paramètres (utilisateurs, groupes, associations) de l'ancienne base vers la nouvelle. Cette option est disponible dans la gestion des mots de passe.

Pour aller plus loin

Vous pouvez (re)créer ou modifier par programmation l'ensemble des mots de passe et attributions.
Vous pouvez également gérer votre propre système de mots de passe. Dans ce cas, vous devrez gérer l'intégralité des implications de telle ou telle ouverture de droits à un utilisateur. Comme toujours, c'est une solution plus souple mais qui nécessite un supplément de programmation pour remplacer ce que 4D fait déjà.

31

Triggers

Objectif(s) pédagogique(s)	<ul style="list-style-type: none">• Paramétrage des événements moteur• Prise en compte des événements moteur dans la programmation
Durée estimée	10 minutes
Ce que nous allons utiliser	Événements moteurs, optimisation des traitements par centralisation

Un trigger (méthode table) sert à contrôler les actions effectuées par le moteur de la base sur les données (création, modification, suppression, chargement). A ce titre, un trigger est une méthode comme une autre. C'est juste son contexte d'exécution qui change.

En effet, les triggers se déclenchent dans le cadre d'événements MOTEUR et non plus d'événements formulaires comme nous l'avons vu jusqu'à présent. Cette appellation d'événement MOTEUR apporte une précision importante : les triggers sont exécutés sur le moteur de la base de données.

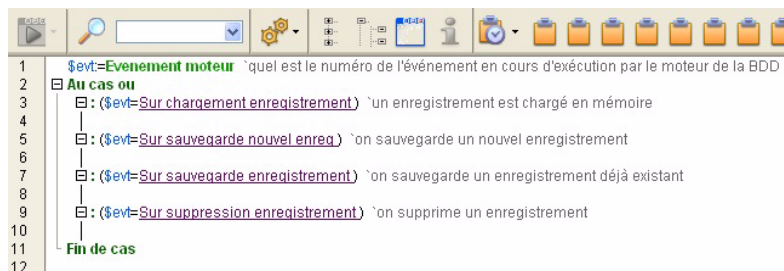
En monoposte, il s'agit bien du programme 4D qui contient à la fois le moteur de base de données, le moteur de rendu (affichages, etc.). En mode client/serveur, les données sont stockées sur le serveur et envoyées aux postes clients par le moteur de 4D Server. Dans ce cas, les triggers sont exécutés sur le serveur et non plus en local sur les postes clients.

MISE EN OEUVRE

A l'instar des événements formulaires, les événements moteurs doivent être cochés pour que 4D passe la main au trigger à exécuter. C'est dans l'Inspecteur de tables que nous pouvons activer tel ou tel événement moteur, comme vous le voyez sur la copie écran suivante :



En bas à droite se trouve le bouton "Editer" qui permet de créer la méthode à exécuter lors de la survenue des événements cochés. La méthode est généralement structurée de la manière suivante :



Contrairement à une méthode formulaire qui n'est exécutée que lorsqu'on est dans le cadre d'un affichage (ou une impression), un trigger est exécuté quelle que soit la manière dont on accède aux enregistrements (affichage dans un formulaire, programmation, import/export, ODBC...). Le trigger a l'avantage d'être un point de passage obligé.

Sachant qu'un trigger bloque le reste des traitements durant son exécution, les principes généraux à respecter dans un trigger sont :

- rapidité d'exécution
- pas d'interface (alertes, messages...)

EXERCICE



Ajoutez un champ *Durée* dans la table *Intervention* et faites en sorte qu'il se calcule automatiquement lorsque l'heure de fin d'intervention est renseignée.



Voir corrigé page 258.

COMMENTAIRES



D'une manière générale, prenez l'habitude de créer des méthodes projet qui réalisent vos traitements, calculs, etc. et appelez ces méthodes "génériques" depuis les méthodes spécifiques telles que les triggers. Nous avons vu comment passer des paramètres alors profitez-en, c'est beaucoup plus facile à maintenir et vous gagnerez un temps considérable tant en développement qu'en correction de bogues. Votre code sera également plus concis, fiable et lisible.

Pour aller plus loin

On peut exécuter des triggers en cascade. Dans ce cas, 4D dispose de commandes permettant de connaître les propriétés du trigger et le niveau dans la cascade.

Un trigger peut également servir à "accepter" ou "rejeter" la tentative d'opération sur l'enregistrement. C'est un moyen qui vous permet de garantir l'intégrité de votre base de données quelle que soit la manière dont les informations sont traitées (ODBC, plug-in, saisie, import...)

Triggers

32

Sélection courante

Objectif(s) pédagogique(s)	<ul style="list-style-type: none"> • Une sélection... c'est quoi ? • Comment constituer une sélection et naviguer d'enregistrement en enregistrement • Interaction entre les sélections et les tableaux
Durée estimée	45 minutes
Ce que nous allons utiliser	Editeur de méthodes, Tableaux, Editeur de formulaires, Listbox, Propriétés des objets

Maintenant que l'interface et les principales fonctionnalités sont en place, nous allons manipuler les enregistrements et découvrir les possibilités de 4D.

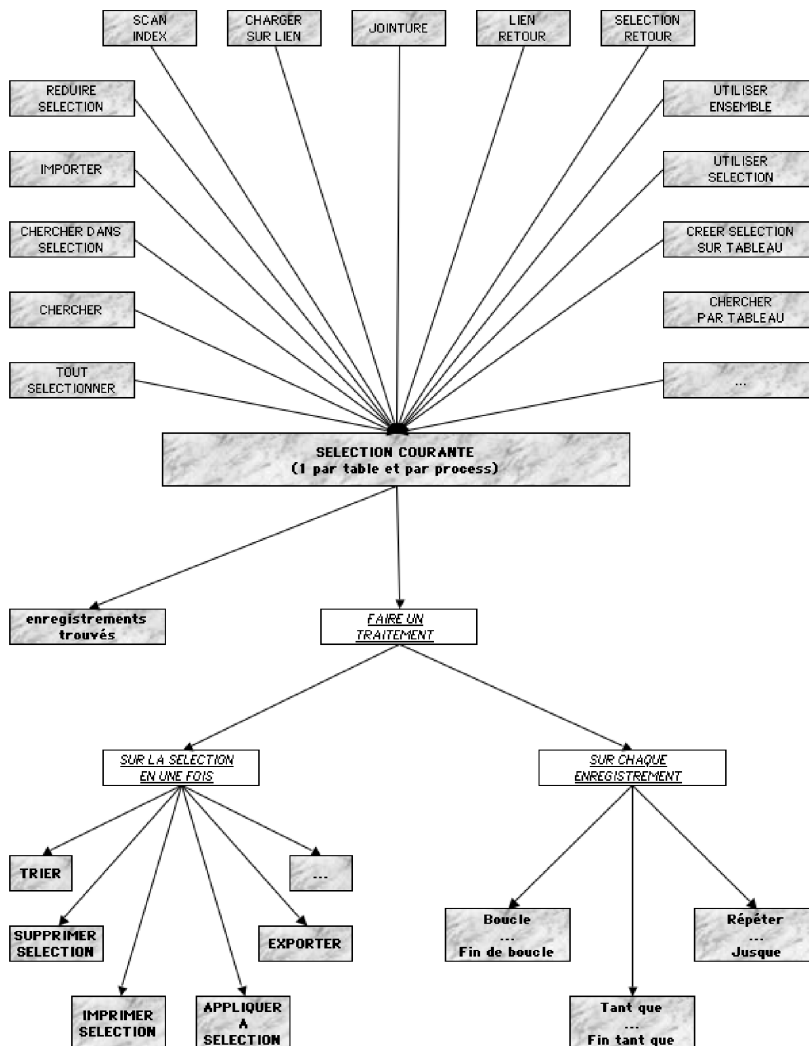
Avant tout, abordons la notion de *Sélection courante*. C'est une des notions fondamentales de 4D qui, bien qu'évidente, diffère quelque peu des concepts habituels notamment en SQL (nous aborderons la partie SQL de 4D plus loin dans les leçons).

Une sélection courante est une liste d'enregistrements obtenue par une recherche sur une table. Ainsi dans 4D, nous avons en permanence une sélection courante par table. Cette sélection peut contenir de 0 à N enregistrements. La sélection courante constitue la liste des enregistrements sur lesquels vont s'appliquer les traitements à venir. On peut changer de sélection courante entre deux traitements.

Prenons un exemple : si je souhaite imprimer les interventions du mois de mai 2007, il faut que je les recherche (je crée une sélection courante) puis que j'imprime ces enregistrements en utilisant par exemple un formulaire sortie qui contient quelques champs de la table interventions (Date, Objet...).

Le schéma ci-dessous représente la manière donc 4D fonctionne, c'est à dire le mode SELECTION-ACTION :

CREER ET UTILISER UNE SELECTION COURANTE



MISE EN OEUVRE

Comme il est indiqué sur le schéma ci-dessus, la commande CHERCHER est un des moyens d'obtenir une sélection courante.

Nous allons la mettre en œuvre dans la réalisation d'un tableau statistique présenté dans une ListBox. Il s'agit d'obtenir le nombre d'interventions par Technicien sur une période donnée.

L'ordre des traitements est le suivant (après avoir déterminé la période souhaitée) :

- 1°) chercher tous les techniciens (1 sélection courante dans la table Techniciens)
- 2°) faire une boucle sur la sélection courante et prendre chaque enregistrement l'un après l'autre
- 3°) pour chaque technicien, retrouver ses interventions (1 sélection courante dans la table Interventions)
- 4°) chercher dans cette sélection les interventions de la période souhaitée
- 5°) stocker le résultat dans un tableau
- 6°) ajouter dans un 2e tableau le nom du technicien
- 7°) afficher les 2 tableaux dans un dialogue

NB : nous verrons une deuxième méthode pour obtenir le même résultat et referons cet exercice en incluant des commandes SQL dans 4D pour que vous voyiez les différentes possibilités.

En attendant au travail ! Reprenons les points dans l'ordre :

Préliminaire : déterminer la période souhaitée. Nous nous limiterons dans cet exercice à l'année en cours du 1er janvier au 31/12. Ensuite, vous pourrez améliorer le système pour présenter à l'utilisateur un choix de périodes.

1 Créez une méthode nommée STATISTIQUES qui intègre 2 variables pour les bornes de dates :

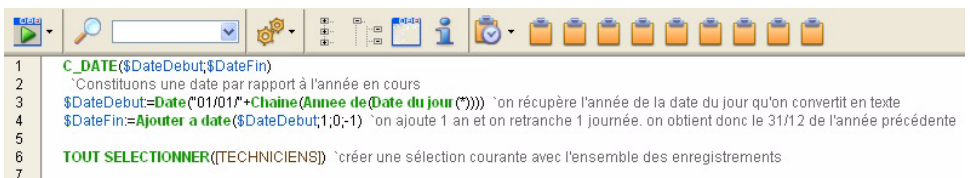


```

1 C_DATE($DateDebut;$DateFin)
2 `Constituons une date par rapport à l'année en cours
3 $DateDebut=Date("01/01"+Chaine(Annee de(Date du jour (*)))) `on récupère l'année de la date du jour qu'on convertit en texte
4 $DateFin=Ajouter a date($DateDebut;1;0;-1) `on ajoute 1 an et on retranche 1 journée. on obtient donc le 31/12 de l'année précédente
5

```

2 Cherchez tous les techniciens (1 sélection courante dans la table Techniciens) :

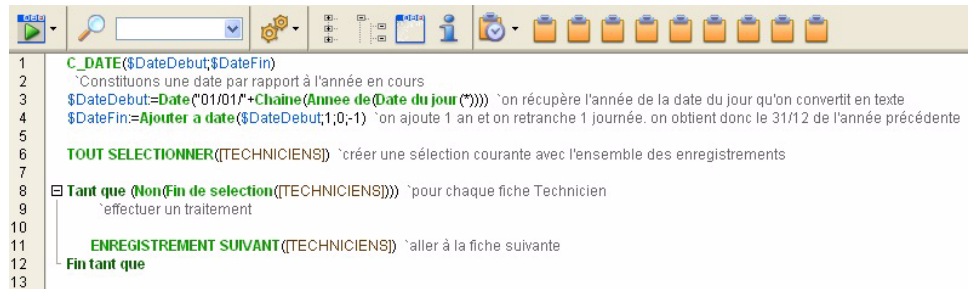


```

1 C_DATE($DateDebut;$DateFin)
2 `Constituons une date par rapport à l'année en cours
3 $DateDebut=Date("01/01"+Chaine(Annee de(Date du jour (*)))) `on récupère l'année de la date du jour qu'on convertit en texte
4 $DateFin=Ajouter a date($DateDebut;1;0;-1) `on ajoute 1 an et on retranche 1 journée. on obtient donc le 31/12 de l'année précédente
5
6 TOUT SELECTIONNER((TECHNICIENS)) `créer une sélection courante avec l'ensemble des enregistrements
7

```

3 Faites une boucle sur la sélection courante et prenez chaque enregistrement l'un après l'autre :

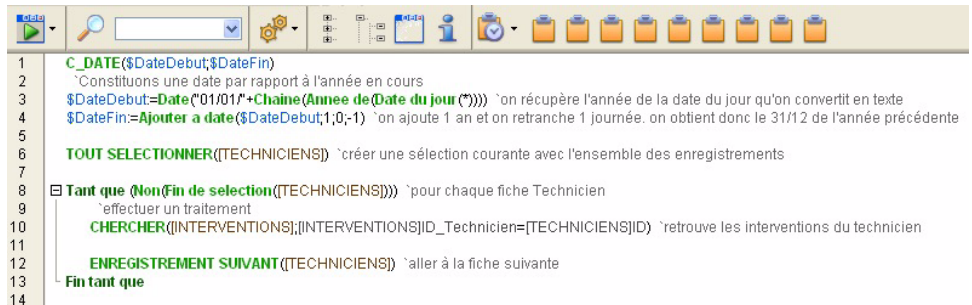


```

1 C_DATE($DateDebut,$DateFin)
2 `Constituons une date par rapport à l'année en cours
3 $DateDebut=Date("01/01/"+Chaine(Annee de(Date du jour (*)))) `on récupère l'année de la date du jour qu'on convertit en texte
4 $DateFin=Ajouter a date($DateDebut;1;0;-1) `on ajoute 1 an et on retranche 1 journée. on obtient donc le 31/12 de l'année précédente
5
6 TOUT SELECTIONNER((TECHNICIENS)) `créer une sélection courante avec l'ensemble des enregistrements
7
8
9  Tant que (Non(Fin de selection((TECHNICIENS)))) `pour chaque fiche Technicien
10 `effectuer un traitement
11
12 ENREGISTREMENT SUIVANT((TECHNICIENS)) `aller à la fiche suivante
13 Fin tant que

```

4 Pour chaque technicien, retrouvez ses interventions (1 sélection courante dans la table Interventions) :

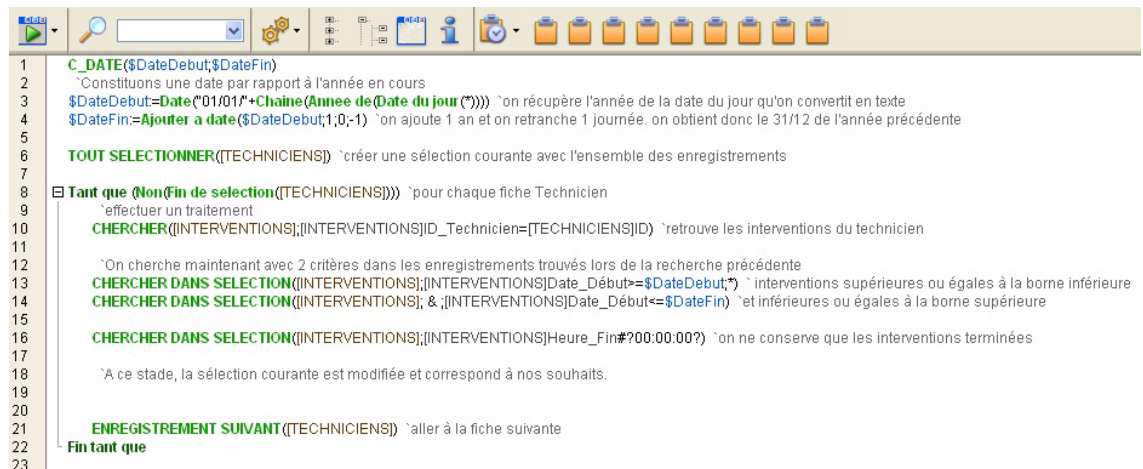


```

1 C_DATE($DateDebut,$DateFin)
2 `Constituons une date par rapport à l'année en cours
3 $DateDebut=Date("01/01/"+Chaine(Annee de(Date du jour (*)))) `on récupère l'année de la date du jour qu'on convertit en texte
4 $DateFin=Ajouter a date($DateDebut;1;0;-1) `on ajoute 1 an et on retranche 1 journée. on obtient donc le 31/12 de l'année précédente
5
6 TOUT SELECTIONNER((TECHNICIENS)) `créer une sélection courante avec l'ensemble des enregistrements
7
8
9  Tant que (Non(Fin de selection((TECHNICIENS)))) `pour chaque fiche Technicien
10 `effectuer un traitement
11
12  Chercher((INTERVENTIONS);[INTERVENTIONS]ID_Technicien=[TECHNICIENS]ID) `retrouve les interventions du technicien
13 ENREGISTREMENT SUIVANT((TECHNICIENS)) `aller à la fiche suivante
14 Fin tant que

```

5 Cherchez dans cette sélection les interventions de la période souhaitée :



```

1 C_DATE($DateDebut,$DateFin)
2 `Constituons une date par rapport à l'année en cours
3 $DateDebut=Date("01/01/"+Chaine(Annee de(Date du jour (*)))) `on récupère l'année de la date du jour qu'on convertit en texte
4 $DateFin=Ajouter a date($DateDebut;1;0;-1) `on ajoute 1 an et on retranche 1 journée. on obtient donc le 31/12 de l'année précédente
5
6 TOUT SELECTIONNER((TECHNICIENS)) `créer une sélection courante avec l'ensemble des enregistrements
7
8
9  Tant que (Non(Fin de selection((TECHNICIENS)))) `pour chaque fiche Technicien
10 `effectuer un traitement
11
12  Chercher((INTERVENTIONS);[INTERVENTIONS]ID_Technicien=[TECHNICIENS]ID) `retrouve les interventions du technicien
13
14 `On cherche maintenant avec 2 critères dans les enregistrements trouvés lors de la recherche précédente
15  Chercher dans selection((INTERVENTIONS);[INTERVENTIONS]Date_Début>=$DateDebut;*) `interventions supérieures ou égales à la borne inférieure
16  Chercher dans selection((INTERVENTIONS); & ,[INTERVENTIONS]Date_Début<=$DateFin) `et inférieures ou égales à la borne supérieure
17
18  Chercher dans selection((INTERVENTIONS);[INTERVENTIONS]Heure_Fin#?00:00:00?) `on ne conserve que les interventions terminées
19
20 `A ce stade, la sélection courante est modifiée et correspond à nos souhaits.
21
22 ENREGISTREMENT SUIVANT((TECHNICIENS)) `aller à la fiche suivante
23 Fin tant que

```

6 Stockez le résultat dans un tableau.

Mais au fait, vous l'avez déclaré votre tableau ? Déclarons-le avant la boucle.

```

1 C_DATE($DateDebut;$DateFin)
2 `Constituons une date par rapport à l'année en cours
3 $DateDebut=Date("01/01/"+Chaine(Annee de(Date du jour (*)))) `on récupère l'année de la date du jour qu'on convertit en texte
4 $DateFin=Ajouter a date($DateDebut;1;0;-1) `on ajoute 1 an et on retranche 1 journée. on obtient donc le 31/12 de l'année précédente
5
6 TOUT SELECTIONNER((TECHNICIENS)) `créer une sélection courante avec l'ensemble des enregistrements
7
8 TABLEAU ENTIER LONG(TabStatsIntervention;Enregistrements trouvés((TECHNICIENS))) `créer un tableau avec autant de lignes que de techniciens
9
10 Tant que (Non(Fin de selection((TECHNICIENS)))) `pour chaque fiche Technicien
11 `effectuer un traitement
12   CHERCHER((INTERVENTIONS);(INTERVENTIONS)ID_Technicien=(TECHNICIENS)ID) `trouve les interventions du technicien
13
14 `On cherche maintenant avec 2 critères dans les enregistrements trouvés lors de la recherche précédente
15   CHERCHER DANS SELECTION((INTERVENTIONS);(INTERVENTIONS)Date_Début>=$DateDebut;*) `interventions supérieures ou égales à la borne inférieure
16   CHERCHER DANS SELECTION((INTERVENTIONS); & ,(INTERVENTIONS)Date_Début<=$DateFin) `et inférieures ou égales à la borne supérieure
17
18   CHERCHER DANS SELECTION((INTERVENTIONS);(INTERVENTIONS)Heure_Fin#?00:00:00?) `on ne conserve que les interventions terminées
19
20   `A ce stade, la sélection courante est modifiée et correspond à nos souhaits.
21
22   $Indice=Numero dans selection((TECHNICIENS)) `indice précisant dans quelle ligne du tableau on doit stocker le résultat
23   TabStatsIntervention{$Indice}=Enregistrements trouvés((INTERVENTIONS))
24
25   ENREGISTREMENT SUIVANT((TECHNICIENS)) `aller à la fiche suivante
26 Fin tant que
27

```

7 Ajoutez dans un 2^e tableau le nom du technicien. Tiens ! il n'est pas déclaré non plus...

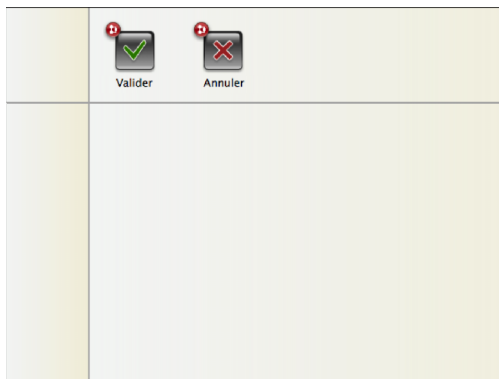
```

1 C_DATE($DateDebut;$DateFin)
2 `Constituons une date par rapport à l'année en cours
3 $DateDebut=Date("01/01/"+Chaine(Annee de(Date du jour (*)))) `on récupère l'année de la date du jour qu'on convertit en texte
4 $DateFin=Ajouter a date($DateDebut;1;0;-1) `on ajoute 1 an et on retranche 1 journée. on obtient donc le 31/12 de l'année précédente
5
6 TOUT SELECTIONNER((TECHNICIENS)) `créer une sélection courante avec l'ensemble des enregistrements
7
8 $Nb=Enregistrements trouvés((TECHNICIENS)) `Combien y a-t-il d'enregistrements dans la sélection
9
10 TABLEAU ENTIER LONG(TabStatsIntervention;$Nb) `créer un tableau avec autant de lignes que de techniciens
11 TABLEAU TEXTE(TabStatsTechniciens;$Nb) `créer un tableau avec autant de lignes que de techniciens
12
13 Tant que (Non(Fin de selection((TECHNICIENS)))) `pour chaque fiche Technicien
14 `effectuer un traitement
15   CHERCHER((INTERVENTIONS);(INTERVENTIONS)ID_Technicien=(TECHNICIENS)ID) `trouve les interventions du technicien
16
17   `On cherche maintenant avec 2 critères dans les enregistrements trouvés lors de la recherche précédente
18   CHERCHER DANS SELECTION((INTERVENTIONS);(INTERVENTIONS)Date_Début>=$DateDebut;*) `interventions supérieures ou égales à la borne inférieure
19   CHERCHER DANS SELECTION((INTERVENTIONS); & ,(INTERVENTIONS)Date_Début<=$DateFin) `et inférieures ou égales à la borne supérieure
20
21   CHERCHER DANS SELECTION((INTERVENTIONS);(INTERVENTIONS)Heure_Fin#?00:00:00?) `on ne conserve que les interventions terminées
22
23   `A ce stade, la sélection courante est modifiée et correspond à nos souhaits.
24
25   $Indice=Numero dans selection((TECHNICIENS)) `indice précisant dans quelle ligne du tableau on doit stocker le résultat
26   TabStatsIntervention{$Indice}=Enregistrements trouvés((INTERVENTIONS))
27   TabStatsTechniciens{$Indice}=[TECHNICIENS]Nom
28
29   ENREGISTREMENT SUIVANT((TECHNICIENS)) `aller à la fiche suivante
30 Fin tant que
31

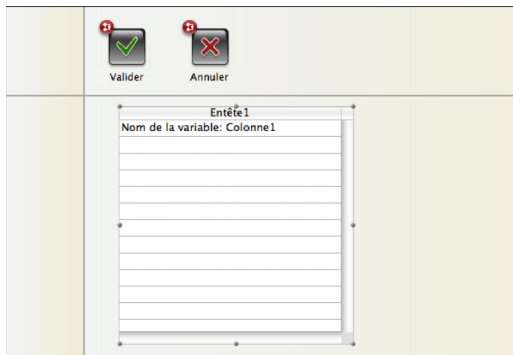
```

Vous devez afficher les 2 tableaux dans un dialogue.

8 Créez un dialogue Projet :



9 Choisissez l'outil List Box et tracez-la dans la partie inférieure droite du formulaire :



Rappel : une Listbox est un objet qui regroupe et synchronise de 1 à N tableaux. Dans une Listbox, vous pouvez paramétrer la Listbox elle-même, chacun des en-têtes de colonnes et chacune des colonnes. En tout, si votre Listbox comporte N colonnes, vous avez $2N+1$ objets (N colonnes, N entêtes, 1 ListBox).

La Listbox permet la saisie de données, le tri et le déplacement des lignes et des colonnes, l'affichage de couleurs alternées. Elle peut être synchronisée avec des tableaux comme nous le faisons ici ou avec des champs de la sélection courante d'une table.

Sachant que la Listbox synchronise ses colonnes, elle prend le plus petit nombre de lignes des tableaux qui la constituent. Ce point est important car il vous arrivera d'avoir des tableaux remplis et une Listbox vide car un seul des tableaux est vide.

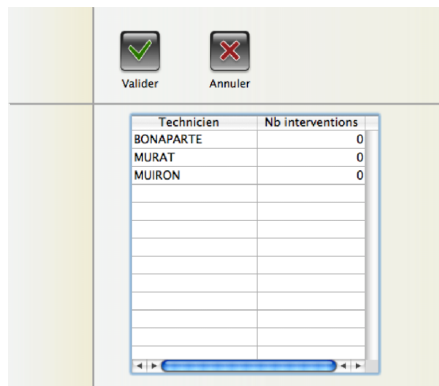
Notre formulaire est terminé. Nous pouvons finaliser la méthode Statistiques en affichant le formulaire :

```

1 C_DATE($DateDebut,$DateFin)
2 *Constituons une date par rapport à l'année en cours
3 $DateDebut=Date("01/01/"+Chaine(Année de(Date du jour (")))) `on récupère l'année de la date du jour qu'on convertit en texte
4 $DateFin=Ajouter a date($DateDebut,1,0,-1) `on ajoute 1 an et on retranche 1 journée. on obtient donc le 31/12 de l'année précédente
5
6 TOUT SELECTIONNER((TECHNICIENS)) `créer une sélection courante avec l'ensemble des enregistrements
7
8 $Nb=Enregistrements trouvés((TECHNICIENS)) `Combien y a-t-il d'enregistrements dans la sélection
9
10 TABLEAU ENTIER LONG(TabStatsIntervention;$Nb) `créer un tableau avec autant de lignes que de techniciens
11 TABLEAU TEXTE(TabStatsTechniciens;$Nb) `créer un tableau avec autant de lignes que de techniciens
12
13
14 Tant que (NonFin de selection((TECHNICIENS))) `pour chaque fiche Technicien
15     *effectuer un traitement
16     CHERCHER((INTERVENTIONS);[INTERVENTIONS]ID_Technicien=[TECHNICIENS]ID) `trouve les interventions du technicien
17
18     *On cherche maintenant avec 2 critères dans les enregistrements trouvés lors de la recherche précédente
19     CHERCHER DANS SELECTION([INTERVENTIONS];[INTERVENTIONS]Date_Début=$DateDebut,*) `interventions supérieures ou égales à la borne inférieure
20     CHERCHER DANS SELECTION([INTERVENTIONS]; & ,[INTERVENTIONS]Date_Début=$DateFin) `et inférieures ou égales à la borne supérieure
21
22     *A ce stade, la sélection courante est modifiée et correspond à nos souhaits.
23
24 $Indice=Numero dans selection((TECHNICIENS)) `indice précisant dans quelle ligne du tableau on doit stocker le résultat
25 TabStatsIntervention($Indice)=Enregistrements trouvés((INTERVENTIONS))
26 TabStatsTechniciens($Indice)=[TECHNICIENS]Nom
27
28 ENREGISTREMENT SUIVANT((TECHNICIENS)) `aller à la fiche suivante
29 Fin tant que
30
31 *créer la fenêtre (conteneur) dans laquelle on affiche le dialogue (contenu)
32 $Fenetre=Creeer fenetre formulaire("STATISTIQUES";Fenêtre standard;Centrée horizontalement;Centrée verticalement)
33 DIALOGUE("STATISTIQUES") `afficher le dialogue
34 FERMER FENETRE($Fenetre) `une fois le dialogue refermé, refermer la fenêtre

```

Et voilà, vous venez de créer un module simple de statistiques !



EXERCICE



Modifiez la programmation de cet exercice pour prendre en compte une 3^e colonne correspondant au temps moyen des interventions.

Dans ce cas, il ne faut prendre que les interventions terminées (dont on possède une heure de fin).



Voir corrigé page 259.

COMMENTAIRES



Pour optimiser ces quelques lignes, on aurait pu dès le départ créer un tableau à partir de la sélection de techniciens avec la commandes VALEURS DISTINCTES, puis faire un boucle sur ce tableau.

On peut réaliser d'autres optimisations avec la commande FIXER DESTINATION RECHERCHE qui évite de changer de sélection. C'est un fonctionnement identique à un SELECT COUNT en SQL.

33

Ensembles et sélections temporaires

Objectif(s) pédagogique(s)	<ul style="list-style-type: none"> • Conserver une sélection • Effectuer des croisements (intersection, réunion...)
Durée estimée	20 minutes
Ce que nous allons utiliser	Editeur de méthodes, programmation et utilisation des sélections temporaires et ensembles

Les ensembles constituent un moyen simple de réaliser des manipulations (réunion, intersection, différence) sur des enregistrements d'une même table. Ils sont un des moyens qui permettent de mettre en attente une sélection pour la réutiliser ultérieurement.

En réalisant une opération sur 2 ensembles, on obtient un ensemble contenant le résultat de l'opération effectuée.

Les ensembles sont constitués d'une série de Bits. Un ensemble contient autant de bits qu'il y a d'enregistrements dans la table. Chaque bit d'un ensemble correspond à l'état "inclus dans la sélection" ou "non inclus" du *n*ème enregistrement de la table au moment où l'ensemble est constitué.

MISE EN OEUVRE

La mise en œuvre est plus simple que l'explication, soyez rassurés :=)

Pour constituer un ensemble, il suffit, à partir d'une sélection courante, d'utiliser la commande **NOMMER ENSEMBLE**. Prenons le cas des interventions commencées avant 09:00:00




```

1 `on recherche les interventions commencées avant 09:00:00
2 CHERCHER([INTERVENTIONS];[INTERVENTIONS]Heure_Début<?09:00:00?)
3 `On constitue un ensemble représentant cette sélection courante
4 NOMMER ENSEMBLE([INTERVENTIONS];"Interventions_Matin")
5

```

Dans cette commande, on indique comme premier paramètre le nom de la table concernée puis le nom que l'on souhaite donner à l'ensemble.

Nous allons créer un 2^e ensemble pour toutes les interventions en cours (avancement < 100%).



```

1 `on recherche les interventions commencées avant 09:00:00
2 CHERCHER([INTERVENTIONS];[INTERVENTIONS]Heure_Début<?09:00:00?)
3 `On constitue un ensemble représentant cette sélection courante
4 NOMMER ENSEMBLE([INTERVENTIONS];"Interventions_Matin")
5
6 `on recherche les interventions dont l'avancement n'est pas à 100%
7 CHERCHER([INTERVENTIONS];[INTERVENTIONS]Avancement<100)
8 `a ce stade, on a perdu la sélection précédente
9
10 `On constitue un ensemble représentant cette nouvelle sélection courante
11 NOMMER ENSEMBLE([INTERVENTIONS];"Interventions_En_Cours")

```

Nous disposons maintenant de deux ensembles que nous pouvons comparer afin d'obtenir :

- les interventions matinales + les interventions en cours
REUNION("Interventions_Matin";"Interventions_En_Cours";"Interventions_Résultat")
- les interventions matinales toujours en cours
INTERSECTION("Interventions_Matin";"Interventions_En_Cours";"Interventions_Résultat")
- les interventions matinales terminées
DIFFERENCE("Interventions_Matin";"Interventions_En_Cours";"Interventions_Résultat")

L'ensemble "Interventions_Résultat" est créé automatiquement par 4D et associé à la table [INTERVENTIONS]. Si nous souhaitons que les enregistrements de l'ensemble résultat deviennent la sélection courante, il suffit d'utiliser cet ensemble :

```
UTILISER ENSEMBLE("Interventions_Résultat")
```

Je sais que vous programmez proprement et prenez soin de votre espace mémoire, donc lorsque vos ensembles ne vous servent plus, vous pouvez les effacer :

```
EFFACER ENSEMBLE("Interventions_Matin")
EFFACER ENSEMBLE("Interventions_En_Cours")
EFFACER ENSEMBLE("Interventions_Resultat")
```

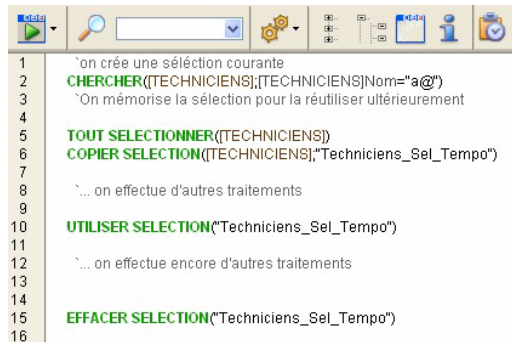
Sélections temporaires

Maintenant que vous connaissez le principe d'utilisation des ensembles, nous allons découvrir la deuxième solution pour conserver une sélection : les *sélections temporaires*.

Le principe d'utilisation est simple, il consiste à utiliser au minimum les deux commandes suivantes :

- **COPIER SELECTION** à l'endroit où vous souhaitez créer une sélection temporaire à partir de votre sélection courante,
- **UTILISER SELECTION** à l'endroit où vous souhaitez régénérer votre sélection courante à partir de la sélection temporaire.

Bien évidemment, lorsque les sélections ne vous servent plus, vous pouvez libérer la mémoire en les effaçant. Voici en résumé la création, l'utilisation et la suppression d'une sélection temporaire :



```
1  "on crée une sélection courante
2  CHERCHER([TECHNICIENS];[TECHNICIENS]Nom="a@")
3  "On mémorise la sélection pour la réutiliser ultérieurement
4
5  TOUT SELECTIONNER([TECHNICIENS])
6  COPIER SELECTION([TECHNICIENS];"Techniciens_Sel_Tempo")
7
8  "... on effectue d'autres traitements
9
10 UTILISER SELECTION("Techniciens_Sel_Tempo")
11
12 "... on effectue encore d'autres traitements
13
14
15 EFFACER SELECTION("Techniciens_Sel_Tempo")
16
```

En termes de mémoire, une sélection nécessite 4 octets par enregistrement de la sélection, quel que soit le nombre d'enregistrements dans la table (contrairement aux ensembles).

Ensembles et sélections temporaires

Le tableau ci-dessous résume les possibilités des ensembles et des sélections temporaires :

Thème	Ensemble	Sélection temporaire
Espace mémoire pour 1 enregistrement	1 bit	4 octets
Conserve le tri	Non	Oui
Conserve l'enregistrement courant	Non	Oui
Réunion	Oui	Non
Intersection	Oui	Non
Différence	Oui	Non
Enregistrer sur disque	Oui	Non
Taille mémoire d'une sélection de 10 enregistrements sur 20 000	20 000 bits soit 2500 octets	10 x 4 octets = 40 octets
Portée	Local, Process, Interprocess	Process, Interprocess

EXERCICE



Pour améliorer l'interface proposée à vos utilisateurs (leur satisfaction n'a pas de prix), vous souhaitez mettre en place le système suivant : partant d'une sélection d'enregistrements, ils doivent pouvoir ajouter de nouveaux enregistrements. Lorsqu'un utilisateur a terminé ses ajouts, il doit retrouver la liste initiale augmentée des nouveaux enregistrements créés.

Vous placerez votre programmation sur le bouton d'ajout et utiliserez (entre autres) les commandes :

- Nommer ensemble
- Adjoindre element
- Reunion
- Utiliser ensemble



Voir corrigé page 261.

COMMENTAIRES



Vous l'aurez deviné, on ne peut comparer que des ensembles d'une même table. Attention : le fonctionnement d'un ensemble implique une utilisation dans un temps limité et éventuellement la pose de *sémaphores* (voir la documentation 4D à ce sujet). En effet, un ensemble fait correspondre un bit à la position physique de chaque enregistrement de la table, or en cas de suppression puis d'ajout d'enregistrements, l'ancien contenu d'un enregistrement physique sera peut-être remplacé par le nouveau contenu qui n'est plus en cohérence avec ce que l'ensemble est censé représenter.

Soyez méthodiques dans l'utilisation des ensembles, qui demeurent un moyen efficace et rapide de comparer des sélections.



ASTUCE

Pour conserver une sélection, une troisième solution consiste à utiliser un tableau qui contient vos identifiants via la commande SELECTION VERS TABLEAU. Vous pouvez utiliser une quatrième solution avec des clusters stockés par exemple dans des BLOBs.

Pour aller plus loin

Maintenant que vous avez compris le principe d'utilisation des ensembles, regardez dans la documentation les autres commandes liées aux ensembles.

34

Process

Objectif(s) pédagogique(s)	<ul style="list-style-type: none"> • Programmation du cycle de vie d'un process : Naissance, vie, mort • Gestion d'une interface spécifique au process
Durée estimée	20 minutes
Ce que nous allons utiliser	Programmation multi-process

Les process sont souvent assimilés à du travail multitâche. Ils permettent de réaliser plusieurs travaux simultanément, sans avoir à terminer le premier traitement pour commencer le deuxième.

Un process est un environnement qui possède son espace mémoire, ses sélections courantes (une par table), ses variables propres (les variables process), éventuellement son interface, etc. Les process peuvent communiquer entre eux de différentes manières (variables interprocess, lecture ou écriture de variables...). Ils peuvent se mettre en attente (process endormis) voire en hibernation (process suspendus) jusqu'à ce qu'un autre process les réveille. Un process meurt seul lorsque la méthode qu'il exécute se termine. On ne peut pas le forcer à mourir par programmation, bien qu'on puisse le programmer pour qu'il meure sous certaines conditions.

Les process servent en général à :

- créer des traitements particuliers (batches, contrôles "sous-marins"),
- créer des palettes d'outils (fenêtre autonomes disposant de fonctions particulières),
- afficher des dialogues de saisie/visualisation pour que l'utilisateur puisse travailler en multifenêtrage.

Certains process sont générés directement par 4D.

MISE EN OEUVRE

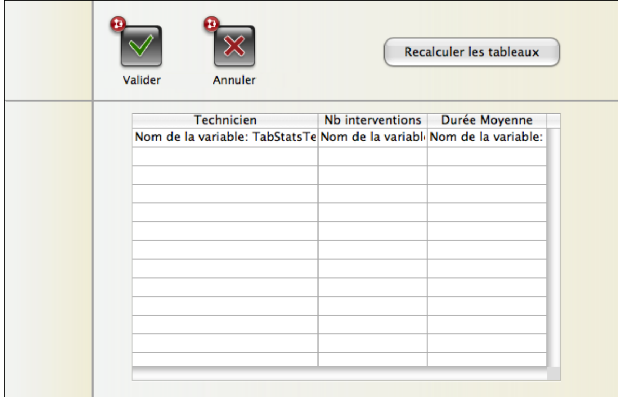
Supposons que vous souhaitiez accéder au module statistique que nous avons mis en place dans les leçons précédentes. Vous voulez qu'il puisse rester actif pendant que vous faites autre chose.

Dans ce cas, nous allons démarrer la méthode dans un process et permettre à l'utilisateur de rafraîchir ses statistiques en cliquant sur un bouton que nous ajouterons dans le formulaire.

Commençons ce dernier point qui consiste à préparer l'interface.

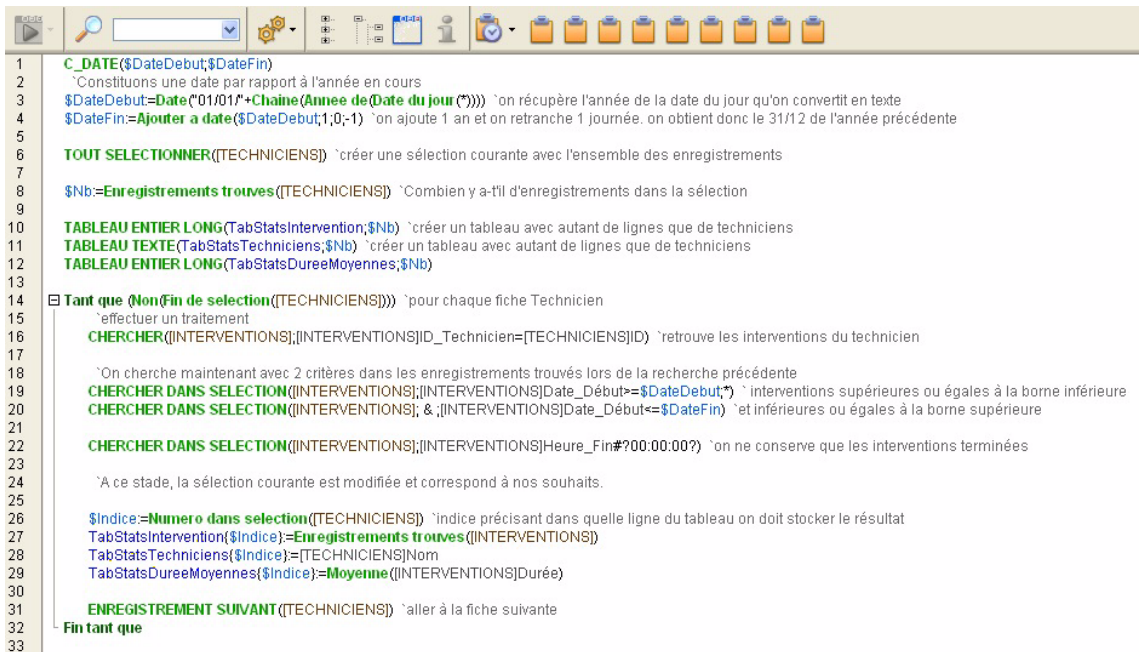
Au cours de l'exercice du chapitre 32, les changements suivants ont été apportés à la base :

- Vous avez créé un bouton qui appelle la méthode de remplissage des tableaux :



The screenshot shows a software interface with a table and control buttons. At the top, there are two buttons: 'Valider' (with a green checkmark icon) and 'Annuler' (with a red X icon), both with a red '+' icon in the top-left corner. To the right of these is a button labeled 'Recalculer les tableaux'. Below the buttons is a table with three columns: 'Technicien', 'Nb interventions', and 'Durée Moyenne'. The first row of the table contains the text 'Nom de la variable: TabStatsTe', 'Nom de la variabl', and 'Nom de la variable:'. The table has approximately 10 rows in total.

- Vous avez déplacé la méthode de recalcul dans le bouton :

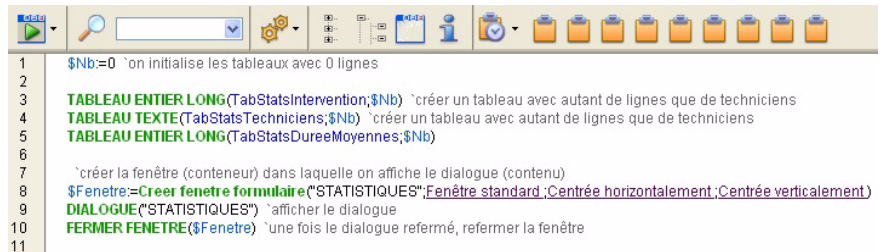


```

1 C_DATE($DateDebut;$DateFin)
2 `Constituons une date par rapport à l'année en cours
3 $DateDebut=Date("01/01/"+Chaine(Annee de(Date du jour(*)))) `on récupère l'année de la date du jour qu'on convertit en texte
4 $DateFin=Ajouter a date($DateDebut;1;0;-1) `on ajoute 1 an et on retranche 1 journée. on obtient donc le 31/12 de l'année précédente
5
6 TOUT SELECTIONNER((TECHNICIENS)) `crée une sélection courante avec l'ensemble des enregistrements
7
8 $Nb:=Enregistrements trouves((TECHNICIENS)) `Combien y a-t'il d'enregistrements dans la sélection
9
10 TABLEAU ENTIER LONG(TabStatsIntervention;$Nb) `créer un tableau avec autant de lignes que de techniciens
11 TABLEAU TEXTE(TabStatsTechniciens;$Nb) `créer un tableau avec autant de lignes que de techniciens
12 TABLEAU ENTIER LONG(TabStatsDureeMoyennes;$Nb)
13
14 Tant que (Non(Fin de selection((TECHNICIENS)))) `pour chaque fiche Technicien
15 `effectuer un traitement
16 CHERCHER((INTERVENTIONS);(INTERVENTIONS)ID_Technicien=(TECHNICIENS)ID) `retrouve les interventions du technicien
17
18 `On cherche maintenant avec 2 critères dans les enregistrements trouvés lors de la recherche précédente
19 CHERCHER DANS SELECTION((INTERVENTIONS);(INTERVENTIONS)Date_Début>=$DateDebut;*) `interventions supérieures ou égales à la borne inférieure
20 CHERCHER DANS SELECTION((INTERVENTIONS); & ,(INTERVENTIONS)Date_Début<=$DateFin) `et inférieures ou égales à la borne supérieure
21
22 CHERCHER DANS SELECTION((INTERVENTIONS);(INTERVENTIONS)Heure_Fin#?00:00:00?) `on ne conserve que les interventions terminées
23
24 `A ce stade, la sélection courante est modifiée et correspond à nos souhaits.
25
26 $indice:=Numero dans selection((TECHNICIENS)) `indice précisant dans quelle ligne du tableau on doit stocker le résultat
27 TabStatsIntervention($indice)=Enregistrements trouves((INTERVENTIONS))
28 TabStatsTechniciens($indice)=(TECHNICIENS)Nom
29 TabStatsDureeMoyennes($indice)=Moyenne((INTERVENTIONS)Durée)
30
31 ENREGISTREMENT SUIVANT((TECHNICIENS)) `aller à la fiche suivante
32 Fin tant que
33

```

- Dans la méthode Statistiques, vous affichez directement le formulaire statistique sans avoir fait les calculs préalables :



```

1 $Nb:=0 `on initialise les tableaux avec 0 lignes
2
3 TABLEAU ENTIER LONG(TabStatsIntervention;$Nb) `créer un tableau avec autant de lignes que de techniciens
4 TABLEAU TEXTE(TabStatsTechniciens;$Nb) `créer un tableau avec autant de lignes que de techniciens
5 TABLEAU ENTIER LONG(TabStatsDureeMoyennes;$Nb)
6
7 `créer la fenêtre (conteneur) dans laquelle on affiche le dialogue (contenu)
8 $Fenetre:=Creat fenetre formulaire("STATISTIQUES";Fenetre standard ;Centrée horizontalement ;Centrée verticalement)
9 DIALOGUE("STATISTIQUES") `afficher le dialogue
10 FERMER FENETRE($Fenetre) `une fois le dialogue refermé, refermer la fenêtre
11

```

Note : Dans cet exemple, je vous propose de conserver l'initialisation des tableaux dans la méthode qui affiche le formulaire. Cette initialisation aurait tout à fait sa place dans la méthode sur chargement du formulaire.

Maintenant, il nous faut créer le process, c'est-à-dire l'environnement d'exécution de cette méthode :

Un process est créé via la commande Nouveau process :

```
vNumeroProcess:=Nouveau process(Méthode;Pile;Nomprocess;
Paramètres;...)
```

Cette ligne de commande doit figurer elle-même dans une méthode. Je vous conseille d'utiliser le principe ci-dessous, vous y gagnerez en temps et en clarté :

La méthode s'appelle elle-même (elle crée une deuxième instance de la méthode dans le nouveau process) et se passe un paramètre :

```

1 `lors de l'appel simple de la méthode elle ne reçoit pas de paramètre
2 Si (Nombre de parametres=0) `si aucun paramètre n'est reçu
3 `nous créons le process
4 C_ENTIER LONG(<=>ProcessStatistiques) `variable contenant le numéro du process
5 <=>ProcessStatistiques:=Nouveau process(Nom methode courante;1024*1024,"Statistiques";"Paramètre fictif";*)
6
7 Simon `si la méthode reçoit au moins un paramètre
8 `elle a probablement été appelée par une autre instance de la même méthode
9 $Nb:=0 `on initialise les tableaux avec 0 lignes
10
11 TABLEAU ENTIER LONG(TabStatsIntervention;$Nb) `créer un tableau avec autant de lignes que de techniciens
12 TABLEAU TEXTE(TabStatsTechniciens;$Nb) `créer un tableau avec autant de lignes que de techniciens
13 TABLEAU ENTIER LONG(TabStatsDureeMoyennes;$Nb)
14
15 `créer la fenêtre (conteneur) dans laquelle on affiche le dialogue (contenu)
16 $Fenetre:=Creer fenetre formulaire("STATISTIQUES";Fenetre standard ;Centrée horizontalement ;Centrée verticalement)
17 DIALOGUE("STATISTIQUES") `afficher le dialogue
18 FERMER FENETRE($Fenetre) `une fois le dialogue refermé, refermer la fenêtre
19
20 Fin de si
21

```

Lors de l'appel de la méthode (sans paramètre), 4D crée un process dans lequel il exécute une méthode. La première méthode se termine tandis que la méthode exécutée par le process continue à se dérouler et présente le tableau de statistiques. Vous pouvez ainsi continuer à travailler tout en ayant la fenêtre de statistiques à portée de main.



Pour tracer l'exécution d'un process (au moment où vous exécutez la ligne Nouveau process), utilisez le bouton "pas à pas nouveau process". Ce bouton exécute la ligne et ouvre une deuxième fenêtre de trace dans laquelle vous pouvez suivre le déroulement de la méthode exécutée dans le process, indépendamment de la méthode appelante.

EXERCICE



Créez un process qui affiche l'heure et le temps écoulé depuis que ce nouveau process est lancé. Cet exercice vous donnera les bases de la mise en oeuvre des compteurs temps.



Voir corrigé page 261.

COMMENTAIRES



Cet exemple simple vous permet d'envisager des évolutions rapides de votre programmation. Vous concevez vos traitements, les testez puis en quelques lignes pouvez les paramétrer pour une exécution autonome dans un process.

Les exemples d'utilisation de process sont nombreux. En voici un autre : vous avez des mises à jour à faire sur une base de données importante, or ces mises à jour peuvent être décalées de quelques secondes, voire minutes sans impact sur la bonne marche de l'entreprise.

Vous pouvez prévoir une table dans laquelle vous enregistrez une liste de tâches à réaliser. Pour chacune des tâches vous précisez une date, une heure, éventuellement un niveau de priorité et un descriptif du traitement à réaliser (quoi, sur quelle table, quel enregistrement...). Ensuite, vous mettez en route un process dont la seule mission est de consulter cette table et de réaliser l'ensemble des tâches qui y figurent. Lorsqu'il a terminé, il se suspend, jusqu'à ce qu'une nouvelle tâche soit enregistrée et que le process soit réactivé.

Pour aller plus loin

Communication Interprocess, verrouillage d'enregistrements, sémaphores

35

SQL

Objectif(s) pédagogique(s)	Effectuer des requêtes SQL dans 4D
Durée estimée	40 minutes
Ce que nous allons utiliser	Mise en oeuvre des commandes SQL simples intégrées à la programmation 4D sous 2 formes : <ul style="list-style-type: none">• plusieurs lignes de commande SQL insérées dans un bloc de traitement SQL• une seule ligne de commande qui exécute la liste de commandes stockées dans une variable texte

Outre le langage natif de 4D, vous pouvez utiliser dans 4D des commandes SQL comme vous le faites dans d'autres outils. Ces commandes peuvent être intégrées dans les développements existants de manière très simple.

Nous reprendrons l'exemple des statistiques pour créer le même tableau à partir de commandes SQL.

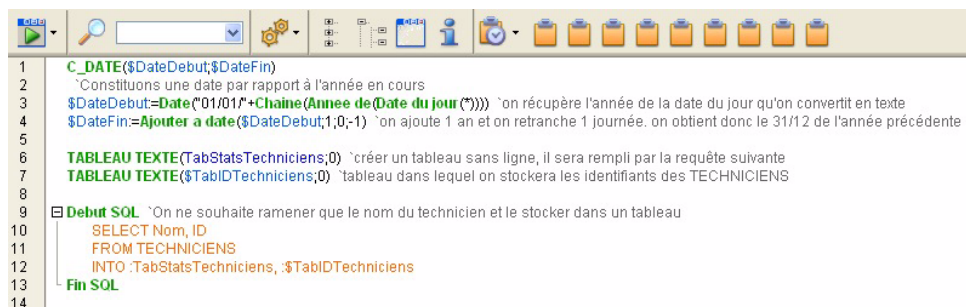
MISE EN OEUVRE

Créez une nouvelle méthode que vous appelez *STATISTIQUES_SQL*.

Nous décomposons la démarche (remplissage des tableaux statistiques) en deux parties :

- obtention de la liste des techniciens,
- pour chaque technicien, calcul de ses statistiques.

Voici l'aspect de la méthode qui permet de réaliser le premier tableau :



```

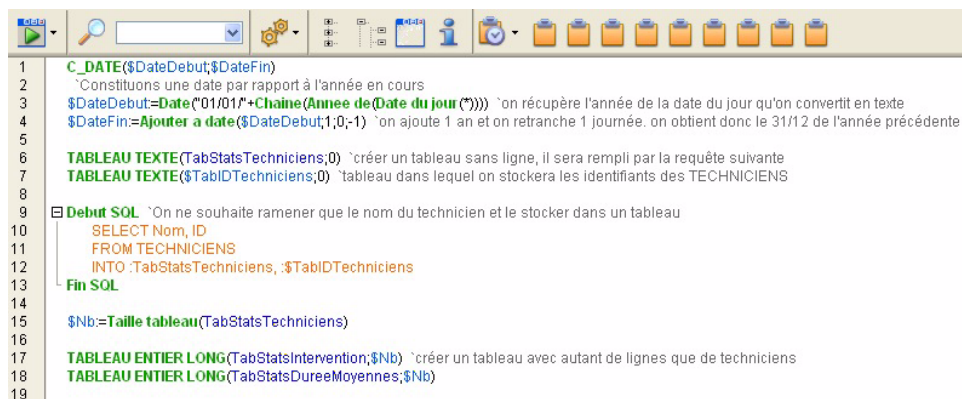
1 C_DATE($DateDebut;$DateFin)
2 `Constituons une date par rapport à l'année en cours
3 $DateDebut:=Date("01/01/"+Chaine(Annee de(Date du jour (*)))) `on récupère l'année de la date du jour qu'on convertit en texte
4 $DateFin:=Ajouter a date($DateDebut;1;0;-1) `on ajoute 1 an et on retranche 1 journée. on obtient donc le 31/12 de l'année précédente
5
6 TABLEAU TEXTE(TabStatsTechniciens;0) `créer un tableau sans ligne, il sera rempli par la requête suivante
7 TABLEAU TEXTE($TabIDTechniciens;0) `tableau dans lequel on stockera les identifiants des TECHNICIENS
8
9 Debut SQL `On ne souhaite ramener que le nom du technicien et le stocker dans un tableau
10 SELECT Nom, ID
11 FROM TECHNICIENS
12 INTO :TabStatsTechniciens, :$TabIDTechniciens
13 Fin SQL
14

```

Vous aurez noté la syntaxe qui indique à 4D de transférer le résultat dans les tableaux. On utilise le nom de la variable précédé du symbole ":" (deux-points).

A ce stade, nous avons 2 tableaux remplis (et synchronisés).

La suite de la méthode (dimensionnement des tableaux) reste globalement structurée de la même manière :



```

1 C_DATE($DateDebut;$DateFin)
2 `Constituons une date par rapport à l'année en cours
3 $DateDebut:=Date("01/01/"+Chaine(Annee de(Date du jour (*)))) `on récupère l'année de la date du jour qu'on convertit en texte
4 $DateFin:=Ajouter a date($DateDebut;1;0;-1) `on ajoute 1 an et on retranche 1 journée. on obtient donc le 31/12 de l'année précédente
5
6 TABLEAU TEXTE(TabStatsTechniciens;0) `créer un tableau sans ligne, il sera rempli par la requête suivante
7 TABLEAU TEXTE($TabIDTechniciens;0) `tableau dans lequel on stockera les identifiants des TECHNICIENS
8
9 Debut SQL `On ne souhaite ramener que le nom du technicien et le stocker dans un tableau
10 SELECT Nom, ID
11 FROM TECHNICIENS
12 INTO :TabStatsTechniciens, :$TabIDTechniciens
13 Fin SQL
14
15 $Nb:=Taille tableau(TabStatsTechniciens)
16
17 TABLEAU ENTIER LONG(TabStatsIntervention;$Nb) `créer un tableau avec autant de lignes que de techniciens
18 TABLEAU ENTIER LONG(TabStatsDureeMoyennes;$Nb)
19

```

Maintenant, nous utilisons encore des commandes SQL pour valoriser les tableaux :

```

1 C_DATE($DateDebut;$DateFin)
2 `Constituons une date par rapport à l'année en cours
3 $DateDebut=Date("01/01/"+Chaine(Annee de(Date du jour (*)))) `on récupère l'année de la date du jour qu'on convertit en texte
4 $DateFin=Ajouter a date($DateDebut;1;0;-1) `on ajoute 1 an et on retranche 1 journée. on obtient donc le 31/12 de l'année précédente
5
6 TABLEAU TEXTE(TabStatsTechniciens;0) `créer un tableau sans ligne, il sera rempli par la requête suivante
7 TABLEAU TEXTE($TabIDTechniciens;0) `tableau dans lequel on stockera les identifiants des TECHNICIENS
8
9 [ Debut SQL `On ne souhaite ramener que le nom du technicien et le stocker dans un tableau
10     SELECT Nom, ID
11     FROM TECHNICIENS
12     INTO :TabStatsTechniciens, $TabIDTechniciens
13   Fin SQL
14
15   $Nb=Taille tableau(TabStatsTechniciens)
16
17   TABLEAU ENTIER LONG(TabStatsIntervention;$Nb) `créer un tableau avec autant de lignes que de techniciens
18   TABLEAU ENTIER LONG(TabStatsDureeMoyennes;$Nb)
19
20 [ Boucle ($i;1;$Nb) `pour chaque ligne du tableau des techniciens
21   $ID=$TabIDTechniciens{$i} `on mémorise l'identifiant à traiter
22   $Result=0 `on initialise la variable pour chaque nouveau technicien
23   C_HEURE($Moyenne)
24
25   [ Debut SQL `on effectue le traitement SQL qui compte le nombre d'interventions
26     SELECT COUNT(*)
27     FROM INTERVENTIONS
28     WHERE INTERVENTIONS.ID_Technicien=$ID
29     INTO :$Result,
30   Fin SQL
31
32   TabStatsIntervention{$i}=$Result `on transfère le résultat dans la ligne de tableau
33
34   `TabStatsDureeMoyennes{$i}=Moyenne((INTERVENTIONS]Duree) `on voit avec cette ligne qu'on peut combiner du code 4D et SQL dans un traitement.
35
36   Fin de boucle
37

```

Ce premier exemple permet de récupérer le nombre total d'interventions réalisées par chaque technicien. Pour limiter le nombre

d'interventions à la période souhaitée, nous pouvons l'écrire comme ceci :

```

1  C_DATE($DateDebut;$DateFin)
2  `Constituons une date par rapport à l'année en cours
3  $DateDebut=Date("01/01/"+Chaine(Annee de(Date du jour(*)))) `on récupère l'année de la date du jour qu'on convertit en texte
4  $DateFin=Ajouter a date($DateDebut;1;0;-1) `on ajoute 1 an et on retranche 1 journée. on obtient donc le 31/12 de l'année précédente
5
6  TABLEAU TEXTE(TabStatsTechniciens;0) `créer un tableau sans ligne, il sera rempli par la requête suivante
7  TABLEAU TEXTE($TabIDTechniciens;0) `tableau dans lequel on stockera les identifiants des TECHNICIENS
8
9  [ Debut SQL `On ne souhaite ramener que le nom du technicien et le stocker dans un tableau
10     SELECT Nom, ID
11     FROM TECHNICIENS
12     INTO :TabStatsTechniciens, :$TabIDTechniciens
13  ] Fin SQL
14
15  $Nb:=Taille tableau(TabStatsTechniciens)
16
17  TABLEAU ENTIER LONG(TabStatsIntervention;$Nb) `créer un tableau avec autant de lignes que de techniciens
18  TABLEAU ENTIER LONG(TabStatsDureeMoyennes;$Nb)
19
20  [ Boucle ($i;1;$Nb) `pour chaque ligne du tableau des techniciens
21     $ID:$TabIDTechniciens{$i} `on mémorise l'identifiant à traiter
22     $Result=0 `on initialise la variable pour chaque nouveau technicien
23     C_HEURE($Moyenne)
24
25     [ Debut SQL `on effectue le traitement SQL qui compte le nombre d'interventions
26         SELECT COUNT(*)
27         FROM INTERVENTIONS
28         WHERE INTERVENTIONS.ID_Technicien=$ID AND INTERVENTIONS.Date_Debut>=$DateDebut AND INTERVENTIONS.Date_Debut<=$DateFin
29         INTO :$Result;
30     ] Fin SQL
31
32     TabStatsIntervention{$i}=$Result `on transfère le résultat dans la ligne de tableau
33
34     `TabStatsDureeMoyennes{$i}=Moyenne([INTERVENTIONS]Duree) `on voit avec cette ligne qu'on peut combiner du code 4D et SQL dans un traitement.
35
36  ] Fin de boucle
37

```

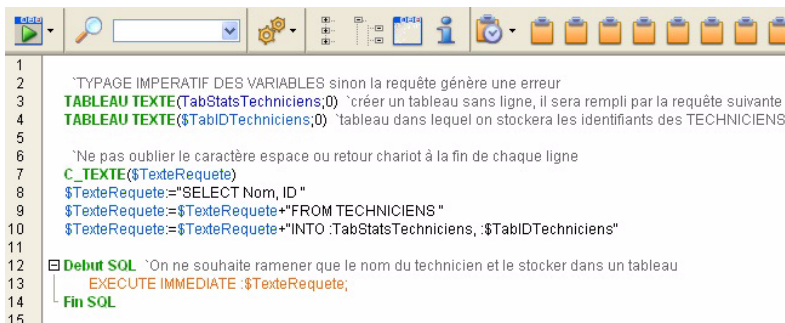
Afin d'éviter une ambiguïté sur le nom des champs, vous pouvez écrire la requête en préfixant les champs du nom de la table :

```

24  [ Debut SQL `on effectue le traitement SQL qui compte le nombre d'interventions
25     SELECT COUNT(*)
26     FROM INTERVENTIONS
27     WHERE INTERVENTIONS.ID_Technicien=$ID AND INTERVENTIONS.Date_Debut>=$DateDebut AND INTERVENTIONS.Date_Debut<=$DateFin
28     INTO :$Result;
29  ] Fin SQL
30
31

```

Une autre manière d'exécuter une commande SQL est de constituer un texte contenant l'ensemble de la requête puis d'exécuter ce texte comme ceci :



```

1
2  *TYPAGE IMPERATIF DES VARIABLES sinon la requête génère une erreur
3  TABLEAU TEXTE(TabStatsTechniciens;0) `créer un tableau sans ligne, il sera rempli par la requête suivante
4  TABLEAU TEXTE($TabIDTechniciens;0) `tableau dans lequel on stockera les identifiants des TECHNICIENS
5
6  `Ne pas oublier le caractère espace ou retour chariot à la fin de chaque ligne
7  C_TEXTE($TexteRequete)
8  $TexteRequete:="SELECT Nom, ID "
9  $TexteRequete:=$TexteRequete+"FROM TECHNICIENS "
10 $TexteRequete:=$TexteRequete+"INTO :TabStatsTechniciens, :$TabIDTechniciens"
11
12  Debut SQL `On ne souhaite ramener que le nom du technicien et le stocker dans un tableau
13 EXECUTE IMMEDIATE :$TexteRequete;
14 Fin SQL
15

```



ASTUCE

On ne peut pas utiliser de pointeurs et de lignes de tableaux dans une requête SQL, l'interpréteur ne les prend pas en compte. Dans ce cas, il est souvent préférable de constituer la requête sous forme de texte et de l'exécuter via la commande EXECUTE IMMEDIATE.

EXERCICE



Exercice 1 : Pour compléter ce tableau statistique, vous allez créer un tableau de synthèse présentant pour l'ensemble des tables le nombre d'enregistrements qu'elles contiennent. Les résultats (chiffrés) peuvent varier par rapport au corrigé, en fonction du nombre d'enregistrements que vous avez créés.

Exercice 2 : Créez un requêteur qui permet de saisir ses propres commandes SQL et de les exécuter en mode utilisateur.



Voir corrigé page 264.

COMMENTAIRES



Les deux langages apportent leur lot de souplesse et de possibilités. Si vous commencez un nouveau développement, vous pouvez décider d'accéder au moteur de la base de données uniquement en SQL et conserver le langage de 4D pour vos autres besoins (l'interface, la gestion du contexte utilisateur, etc.)



Si vous reprenez un développement existant, vous pouvez remplacer au fur et à mesure une partie de la programmation par des commandes SQL, ce sera un très bon entraînement. A ce titre, je vous conseille d'encapsuler votre code dans un test simple comme celui-ci :

```
▼ Si (◊EN_TEST) `seulement sur votre poste cette variable ◊EN_TEST est à VRAI.  
`vous passez dono dans cette partie de la programmation  
  
`... ici figurent vos nouvelles lignes de programme  
`... les autres ne les verront que lorsque vous les aurez validées  
  
▼ Sinon `tant que votre nouvelle programmation n'est pas validée  
`vous conservez l'ancienne programmation qui fonctionne :=))  
  CHERCHER($Table->,$Champ->=>$NumRef)  
  MODIFIER ENREGISTREMENT($Table->,*  
  Fin de si
```

Pour aller plus loin

Découvrez les 4 manières d'utiliser le SQL dans vos développements

36

BLOBS

Objectif(s) pédagogique(s)	<ul style="list-style-type: none"> • Comprendre et programmer les BLOBs • Interaction des BLOBs avec les variables • Enregistrement et lecture d'un BLOB dans un document
Durée estimée	15 minutes
Ce que nous allons utiliser	BLOBs, variables, débogueur

Les BLOBs permettent de stocker et récupérer toutes sortes d'informations.

Au début, on se pose toujours les questions suivantes :

- A quoi ça peut bien me servir et dans quelles circonstances ?
- Mettre des "choses" dedans je veux bien, mais comment et surtout dans quel ordre je vais les ressortir ?

Répondons d'abord à la deuxième question : bien qu'on puisse accéder à n'importe quel octet du BLOB, dans la plupart des traitements vous ferez du FIFO (First In First Out) c'est à dire que vous récupérerez le contenu du BLOB dans l'ordre dans lequel vous l'avez rangé.

Pour faire un parallèle simple, prenez une étagère dans laquelle vous rangez des livres de gauche à droite. Vous commencez par ranger un petit livre à gauche puis un gros livre collé au premier puis encore un gros livre puis une photo de votre belle-mère puis un bibelot rapporté de vacances, etc. Lorsqu'il faut faire les poussières sur cette étagère, vous ne vous posez pas la question de savoir à combien de centimètres du bord gauche est rangé tel ou tel livre. Vous les prenez dans l'ordre et vous décalez votre regard et vos mains au fur et à mesure, en fonction de la taille du dernier objet enlevé.

Dans un BLOB, c'est pareil. Notamment parce que vous savez ce que vous rangez dans le BLOB. Si vous rangez un entier c'est 2 octets, un

entier long, ce sera 4, une date 6 octets. Pour un texte, vous indiquerez la manière dont vous voulez qu'il soit rangé et comment est indiquée la longueur du texte (chaîne C, chaîne Pascal, etc.).

A la relecture, vous reprendrez les informations dans le même ordre et les rangerez dans des variables adaptées au contenu attendu (typées correctement). Donc la lecture du BLOB se fera de manière cohérente.

Voici maintenant la réponse à la première question (à quoi peut servir un BLOB ?). Vous pouvez les utiliser par exemple dans les cas suivants :

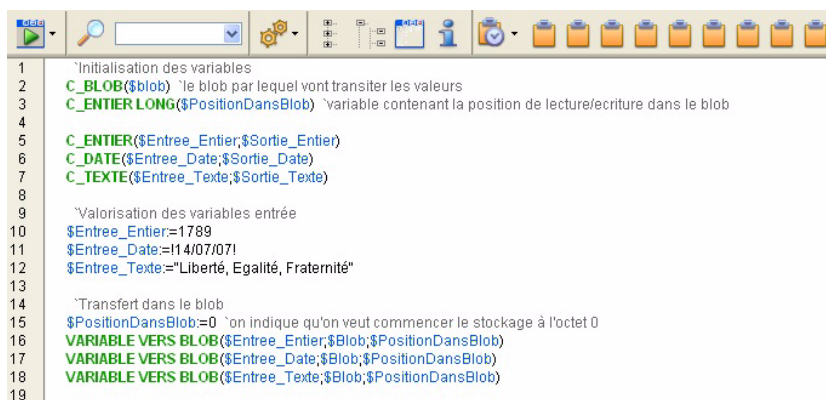
- Stockage de variables, liste hiérarchiques, tableaux
- Echange de documents entre le client et le serveur
- Protection de certaines données confidentielles dans un blob crypté
- Traitement de documents ou de textes de plus de 32K
- Enregistrement des variables contenant des plugins
- Envoi d'images, textes ou documents vers un navigateur web
- Communication avec un web service

Maintenant que vous êtes rassurés... mais dubitatifs voyons par la pratique la simplicité en question.

MISE EN OEUVRE

Prenons le cas simple de stockage de variables. Nous allons valoriser trois variables, les stocker dans un blob puis les récupérer dans trois autres variables de types identiques. Au fur et à mesure, nous contrôlerons en mode trace le remplissage du blob et des variables.

1 Créez la méthode BLOB_GESTION comme ceci :



```

1  `Initialisation des variables
2  C_BLOB($blob) `le blob par lequel vont transiter les valeurs
3  C_ENTIER LONG($PositionDansBlob) `variable contenant la position de lecture/écriture dans le blob
4
5  C_ENTIER($Entree_Entier,$Sortie_Entier)
6  C_DATE($Entree_Date,$Sortie_Date)
7  C_TEXTE($Entree_Texte,$Sortie_Texte)
8
9  `Valorisation des variables entrée
10 $Entree_Entier:=1789
11 $Entree_Date:=11/4/07/07!
12 $Entree_Texte:="Liberté, Egalité, Fraternité"
13
14 `Transfert dans le blob
15 $PositionDansBlob:=0 `on indique qu'on veut commencer le stockage à l'octet 0
16 VARIABLE VERS BLOB($Entree_Entier,$Blob,$PositionDansBlob)
17 VARIABLE VERS BLOB($Entree_Date,$Blob,$PositionDansBlob)
18 VARIABLE VERS BLOB($Entree_Texte,$Blob,$PositionDansBlob)
19

```


La commande VARIABLE VERS BLOB stocke les données dans un format interne 4D, c'est la raison pour laquelle l'espace requis est légèrement supérieur au volume brut de données. L'avantage de cette commande est qu'elle vous évite de gérer le Byte Swapping (position de l'octet de poids fort) en cas de travail multi plate-forme.

Suite à l'exécution de la ligne 16, le BLOB contient les informations suivantes :

The screenshot shows the 'BLOB_GESTION' trace window. It is divided into several sections:

- Objets courants:** A tree view on the left showing the current state of objects like 'Objets courants', 'Variables', 'Constantes', etc.
- Chaine d'appel:** A tree view on the right showing the call stack, with 'BLOB_GESTION' at the top.
- Tableau de valeurs:** A table below the call stack showing the values of various expressions:

Expression	Valeur
\$Entree_Entier	1789
\$Entree_Date	14/07/07
\$Entree_Texte	"Liberté, Egalité, Fraternité"
\$Blob	9 octets
\$Blob(0)	82
\$Blob(1)	86
\$Blob(2)	76
\$Blob(3)	66
\$Blob(4)	9
\$Blob(5)	253
\$Blob(6)	6
\$Blob(7)	0
- Code de la commande:** The bottom section shows the execution steps:


```

*Initialisation des variables
C_BLOB($blob) *le blob par lequel vont transiter les valeurs
C_ENTIER LONG($PositionDansBlob) *variable contenant la position de lecture/ecriture dans le blob

C_ENTIER($Entree_Entier,$Sortie_Entier)
C_DATE($Entree_Date,$Sortie_Date)
C_TEXTE($Entree_Texte,$Sortie_Texte)

*Valorisation des variables entrée
$Entree_Entier:=1789
$Entree_Date:=14/07/07
$Entree_Texte:="Liberté, Egalité, Fraternité"

*Transfert dans le blob
$PositionDansBlob:=0 *on indique qu'on veut commencer le stockage à l'octet 0
VARIABLE VERS BLOB($Entree_Entier,$Blob,$PositionDansBlob)
VARIABLE VERS BLOB($Entree_Date,$Blob,$PositionDansBlob)
VARIABLE VERS BLOB($Entree_Texte,$Blob,$PositionDansBlob)
            
```

Le stockage de la date ajoute les octets suivants dans le BLOB :

\$Blob(9)	82
\$Blob(10)	86
\$Blob(11)	76
\$Blob(12)	66
\$Blob(13)	4
\$Blob(14)	14
\$Blob(15)	0
\$Blob(16)	7
\$Blob(17)	0
\$Blob(18)	215
\$Blob(19)	7

et ainsi de suite. A la fin du troisième ajout, le BLOB contient 85 octets. Maintenant, nous pouvons extraire les données de ce BLOB et les transférer dans les variables de sortie :

Expression	Valeur
\$Entree_Entier	1789
\$Entree_Date	14/07/07
\$Entree_Texte	"Liberté, Egalité, Fraternité"
\$Blob	85 octets
\$Sortie_Entier	1789
\$Sortie_Date	14/07/07
\$Sortie_Texte	==

```

C_TEXTE($Entree_Texte;$Sortie_Texte)

*Valorisation des variables entrée
$Entree_Entier:=1789
$Entree_Date:=14/07/07!
$Entree_Texte:="Liberté, Egalité, Fraternité"

*Transfert dans le blob
$PositionDansBlob:=0 *on indique qu'on veut commencer le stockage à l'octet 0
VARIABLE VERS BLOB($Entree_Entier;$Blob;$PositionDansBlob)
VARIABLE VERS BLOB($Entree_Date;$Blob;$PositionDansBlob)
VARIABLE VERS BLOB($Entree_Texte;$Blob;$PositionDansBlob)

*Extraction des données du blob
$PositionDansBlob:=0 *on indique qu'on veut commencer le stockage à l'octet 0
• BLOB VERS VARIABLE($Blob;$Sortie_Entier;$PositionDansBlob)
  BLOB VERS VARIABLE($Blob;$Sortie_Date;$PositionDansBlob)
  BLOB VERS VARIABLE($Blob;$Sortie_Texte;$PositionDansBlob)

```

Après l'exécution des 2 première lignes, on voit que la récupération des variables est en cours.

Vous voyez qu'il est très simple d'utiliser un BLOB, même si cet exemple en est une approche très limitée.



ASTUCE

*Lorsque vous remplissez un BLOB, vous pouvez utiliser le caractère * à la place de la variable \$PositionDansBlob que nous avons utilisée. 4D comprend avec ce paramètre qu'il doit stocker la variable à la fin du BLOB et l'agrandir en conséquence.*

EXERCICE



Imaginons maintenant que dans votre base de données, vous ayez un certain nombre de paramètres dont il faut tenir compte pour chaque utilisateur (ou chaque site d'installation de votre application) : l'écran de démarrage, la couleur des fonds d'écran, la police et la taille des caractères... Il s'agit en fait des préférences utilisateurs.

Pour les conserver, il existe plusieurs solutions (créer une table, un fichier texte, un fichier XML, etc.). Dans notre cas, nous allons conserver ces préférences dans un BLOB, lui-même stocké dans un fichier sur le disque (les commandes d'écriture et de lecture des BLOBs sont BLOB VERS DOCUMENT et DOCUMENT VERS BLOB).

Créez et écrivez la méthode projet *BLOB_PREFERENCES* qui traite la lecture et l'écriture des variables dans le BLOB ainsi que la lecture et l'écriture du BLOB sur le disque.



Voir corrigé page 266.

COMMENTAIRES



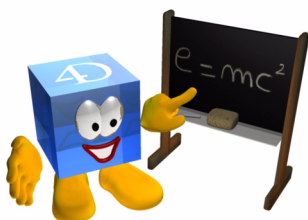
Maintenant que vous maîtrisez les BLOBs, vous allez pouvoir les utiliser dans les différents cas évoqués au début de ce chapitre. Vous pourrez également affiner le fonctionnement en utilisant les autres commandes de lecture/écriture de variables dans les BLOBs. Attention cependant à bien tenir compte du *byteswapping* (la position de l'octet de poids fort est différente entre Macintosh et Windows). En effet, si vous stockez par exemple un entier long sur Windows et le récupérez sur Macintosh, l'ordre des octets sera inversé... gare aux surprises ! Pensez à indiquer l'ordre que vous souhaitez lors du rangement en utilisant le paramètre "ordreOctet".

Pour aller plus loin

Pour gagner de la place et/ou des temps de transfert, pensez également à compresser vos BLOBs. Attention, 4D ne compresses les BLOBs que si leur taille est supérieure à 255 octets. Lors de la décompression de vos BLOBs, pensez à tester si le BLOB est compressé, sinon la tentative de décompression générera une erreur.

37

Corrigés



Tout au long de ce manuel, des exercices permettant d'approfondir la leçon vous sont proposés. Dans la plupart des cas, vous pourrez avoir besoin de vérifier si vos hypothèses sont justes et si ce que vous avez réalisé correspond à ce qui était demandé.

Cette section regroupe tous les "corrigés" des exercices de ce manuel. Chaque exercice est identifié par le nom de la leçon et le numéro de page de l'exercice.

EXERCICE : "FORMULAIRES ET BOUTONS", PAGE 28

Dans chacun des boutons, la méthode affiche la même alerte. Vous ne devez changer que le texte de l'alerte pour l'adapter au bouton concerné.

Vous pouvez modifier le nom d'un bouton en cliquant deux fois sur son titre (attention, ce n'est pas un double clic). Ensuite, tapez le titre et validez la modification par la touche **Entrée** du pavé numérique.

Positionner les boutons : lorsque vous glissez vos boutons, une grille magnétique (traits pointillés rouges) indique la position relative aux autres objets, ainsi que la distance optimale par rapport aux autres objets ou aux bords de la fenêtre. Cette fonctionnalité s'adapte à l'environnement de développement.

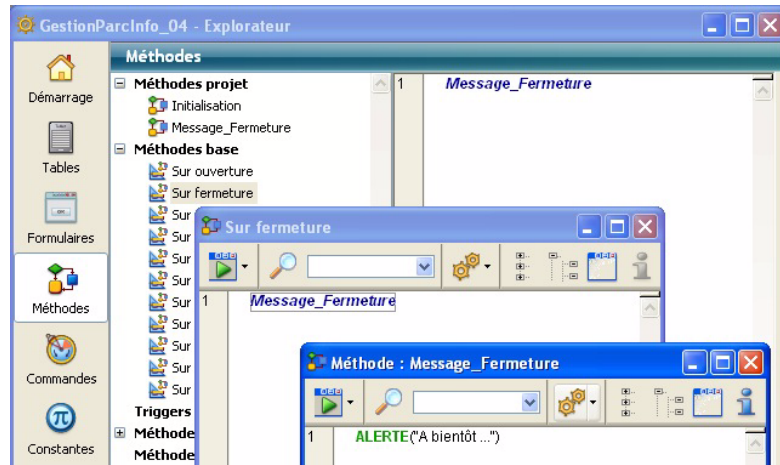
EXERCICE : "AFFICHAGE AU DÉMARRAGE", PAGE 33

Vous avez certainement créé une méthode "Message_Fermeture" qui contient la ligne de code suivante :

```
ALERTE("A bientôt...").
```

Ensuite, vous avez fait appel à cette méthode "Message_Fermeture" dans la méthode base "Sur fermeture".

Sur l'écran suivant, vous voyez l'enchaînement à mettre en place :



ASTUCE

Pour accéder rapidement à une méthode dont le nom figure dans une autre méthode, sélectionnez son nom (en double-cliquant dessus par exemple) puis utilisez le raccourci Ctrl-K pour l'ouvrir directement.

EXERCICE : "PAGES DE FORMULAIRE ET NAVIGATION", PAGE 49

Le premier bouton doit contenir le code ALLER A PAGE(1),
le deuxième ALLER A PAGE(2)
le deuxième ALLER A PAGE(3)
et ainsi de suite.

EXERCICE : "TABLES ET CHAMPS", PAGE 54

Vous devez avoir maintenant les 3 tables principales :

- Techniciens
- Interventions
- Matériels

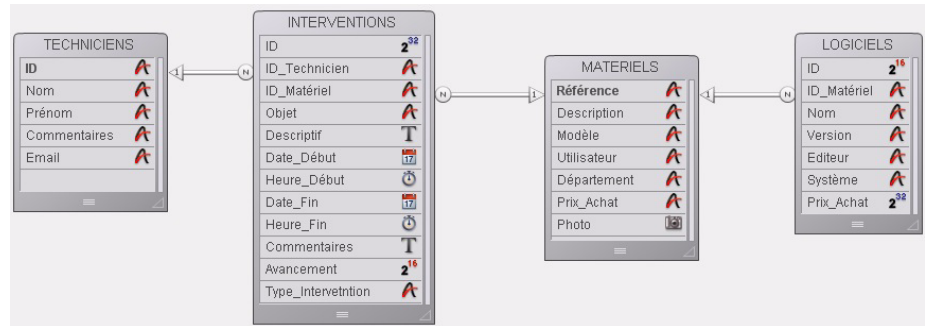
EXERCICE : "SAISIE ET MODIFICATION D'ENREGISTREMENTS", PAGE 61

Vous devez obtenir le résultat suivant :

ID :	Nom :	Version :	Editeur :	Système :
1	OpenOffice	2.1	OpenOffice.org	Windows VISTA
2	NeoOffice	2.1	NeoOffice.org	MacOS X
3	FireFox	2.0	Mozilla.org	Windows VISTA

EXERCICE : "LIENS", PAGE 68

Votre structure doit ressembler à ceci :



EXERCICE : "SAISIE ET SUPPRESSION", PAGE 72

Vous ne devez plus avoir d'enregistrements dans la base.

EXERCICE : "IMPORTER", PAGE 77

Nombre d'enregistrements devant figurer dans vos tables après l'import :

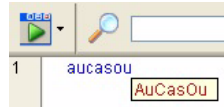
- Enumérations : 27
- Matériels : 72
- Logiciels : 104
- Interventions : 203
- Techniciens : 7

EXERCICE : "FORMULAIRES DE SORTIE", PAGE 85

Lors de la navigation de page en page, vous devez voir défiler les différents formulaires que vous avez positionnés et constater que les sélections d'enregistrements sont bien effectives (rappel de l'ensemble des enregistrements).

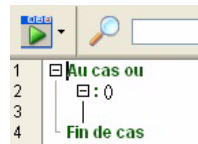
EXERCICE : "RECHERCHES ET TRIS", PAGE 91

Pour gagner du temps, tapez "aucasou" (sans espace) : l'info-bulle vous

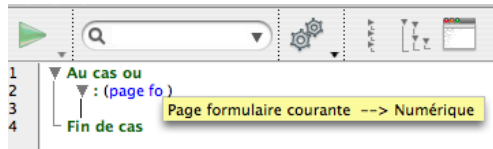


indique que le mot est reconnu.

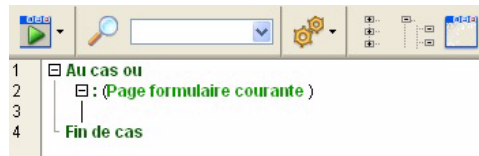
Puis appuyez sur **Tabulation** : 4D écrit automatiquement le texte ci-dessous et positionne le curseur au bon endroit



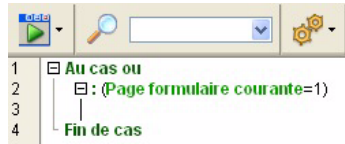
Commencez à taper "page fo" : 4D affiche le nom de la commande qu'il reconnaît et la valeur qu'elle retourne :



Appuyez sur **Tabulation** : le texte s'écrit seul.



Il n'y a plus qu'à compléter avec "=1" :



EXERCICE : "ÉTATS RAPIDES ET EXPORT", PAGE 101

- Liste des interventions par technicien :

Nom	Objet	Date_Début
BERTHIER	Demande Utilis ateur	01/03/2007 00:00:00
BERTHIER	Demande Utilis ateur	30/06/2007 00:00:00
BERTHIER	Demande Utilis ateur	25/05/2007 00:00:00
BERTHIER		
BONAPARTE	Installation	07/07/2007 00:00:00
BONAPARTE	Demande Utilis ateur	13/08/2007 00:00:00
BONAPARTE	Formation	13/07/2007 00:00:00
BONAPARTE	Demande Utilis ateur	31/12/2006 00:00:00
BONAPARTE	Demande Utilis ateur	16/05/2007 00:00:00
BONAPARTE	Installation	30/12/2006 00:00:00
BONAPARTE	Installation	15/04/2007 00:00:00
BONAPARTE	Demande Utilis ateur	23/07/2007 00:00:00
BONAPARTE	Récupération donn ées	06/07/2007 00:00:00
BONAPARTE	Demande Utilis ateur	07/07/2007 00:00:00
BONAPARTE	Demande Utilis ateur	04/01/2007 00:00:00
BONAPARTE	Mise à jour logiciel	22/02/2007 00:00:00
BONAPARTE	Demande Utilis ateur	14/06/2007 00:00:00
BONAPARTE	Demande Utilis ateur	31/08/2007 00:00:00
BONAPARTE	Installation	03/02/2007 00:00:00
BONAPARTE	Demande Utilis ateur	21/02/2007 00:00:00
BONAPARTE	Déblocage utilis ateur	27/06/2007 00:00:00
BONAPARTE	Installation	29/09/2007 00:00:00
BONAPARTE	Installation	03/02/2007 00:00:00
BONAPARTE	Demande Utilis ateur	10/04/2007 00:00:00
BONAPARTE	Récupération donn ées	01/07/2007 00:00:00
BONAPARTE	Installation	30/06/2007 00:00:00
BONAPARTE	Installation	26/04/2007 00:00:00
BONAPARTE	Demande Utilis ateur	04/02/2007 00:00:00
BONAPARTE	Demande Utilis ateur	25/05/2007 00:00:00
BONAPARTE	Demande Utilis ateur	11/03/2007 00:00:00
BONAPARTE	Installation	24/02/2007 00:00:00
BONAPARTE	Installation	11/01/2007 00:00:00
BONAPARTE	Installation	26/09/2007 00:00:00
BONAPARTE	Récupération donn ées	31/03/2007 00:00:00
BONAPARTE	Demande Utilis ateur	26/08/2007 00:00:00
BONAPARTE	Mise à jour système	17/06/2007 00:00:00
BONAPARTE		

■ Nombre d'interventions par technicien :

The screenshot shows an Excel window titled "Etats rapides" with a pivot table and its options. The pivot table has the following structure:

Intitulé	[TECHNICIENS]Nom	[INTERVENTIONS]Objet
Intitulé	Nom	Objet
Détail		
Sous total eur [TECHNICIENS]Nom #		Nombre
Total général		

The "Paramètres de l'état" (PivotTable Options) task pane is open, showing the "Table principale" (Main Table) as "INTERVENTIONS" and "Nouvelles recherches" (New searches) as "203 enreg. dans sélection" and "203 enreg. dans Table". The "Type d'état" (PivotTable type) is set to "Liste" (List). The "Les tables liées" (Related tables) list includes "[TECHNICIENS]". The "Ordre de tri" (Sort order) is set to "[TECHNICIENS]Nom".

Nom	Objet
BERTHIER	44
BONAPARTE	32
LANNES	15
MASSENA	28
MUIRON	27
MURAT	42
NEY	15

- Nombre d'interventions par type d'intervention et technicien :

The screenshot shows the 'Etats rapides' (Quick Reports) window in Microsoft Access. The main area displays a pivot table with the following structure:

Intitulé	[TECHNICIENS]Nom	[INTERVENTIONS]Type_Intervention	[INTERVENTIONS]Objet
Intitulé	Nom	Type Intervention	Objet
Détail			
Sous total sur [INTERVENTIONS]Type_Intervention		#	Nombre
Sous total sur [TECHNICIENS]Nom	#		Nombre
Total général			Nombre

Below the pivot table, the 'Paramètres de l'état' (Report Settings) task pane is visible. It includes:

- Table principale:** INTERVENTIONS
- Nouvelle recherche:** 203 enreg. dans sélection, 203 enreg. dans table
- Type d'état:** Liste (selected), Tableau croisé
- Ordre de tri:** [TECHNICIENS]Nom, [INTERVENTIONS]Type_Intervention
- La table principale:** ID, ID_Technicien, Objet, Descriptif, Date_Début, Heure_Début, Date_Fin, Heure_Fin, Commentaires

Nom	Type_Intervention	Objet
	Formation	11
	Logiciel	4
	Matériel	12
	Périphériques	7
	Réseau	10
BERTHIER		44
	Formation	4
	Logiciel	5
	Matériel	6
	Périphériques	9
	Réseau	8
BONA PARTE		32
	Formation	2
	Matériel	6
	Périphériques	3
	Réseau	4
LANNES		15
	Formation	6
	Logiciel	5
	Matériel	5
	Périphériques	7
	Réseau	5
MAS SENA		28
	Formation	5
	Logiciel	3
	Matériel	7
	Périphériques	8
	Réseau	4
MUIRON		27

- Nombre d'interventions par objet et par technicien :

The screenshot shows the 'Etats rapides' (Quick Reports) window in Microsoft Access. The report is titled 'INTERVENTIONS' and displays data for 203 records. The columns are: 'Initialement' (with a dropdown menu), 'Initialement' (with a dropdown menu), 'Détail' (with a dropdown menu), 'Sous total sur [INTERVENTIONS]Objet' (with a dropdown menu), and 'Sous total sur [TECHNICIENS]Nom' (with a dropdown menu). The 'Total général' row shows a total count of 203. The report is displayed in a 'Tableau croisé' (Crosstab) view. The 'Paramètres de l'état' (Report Parameters) pane shows the 'Table principale' (Main Table) set to 'INTERVENTIONS' and the 'Ordre de tri' (Sort Order) set to 'Ascendant' (Ascending). The 'Table principale' pane shows the fields: ID, ID_Technicien, Objet, Descriptif, Date_Début, Heure_Début, Date_Fin, Heure_Fin, and Commentaires. The 'Ordre de tri' pane shows the fields: [TECHNICIENS]Nom and [INTERVENTIONS]Objet.

Nom	Objet	Objet
	Déblocage utilisateur	3
	Demande Utilisateur	22
	Formation	2
	Installation	12
	Mise à jour logiciel	2
	Mise à jour système	1
	Récupération données	2
BERTHIER		44
	Déblocage utilisateur	1
	Demande Utilisateur	34
	Formation	1
	Installation	11
	Mise à jour logiciel	1
	Mise à jour système	1
	Récupération données	3
BONAPARTE		32
	Demande Utilisateur	6
	Formation	1
	Installation	2
	Mise à jour logiciel	1
	Mise à jour système	2
LANNES		15
	Déblocage utilisateur	1
	Demande Utilisateur	10
	Formation	1
	Installation	12
	Mise à jour logiciel	2
	Mise à jour système	1
	Récupération données	1
MASSENA		28

EXERCICE : "RECHERCHE ET TRI PAR FORMULE, APPLIQUER UNE FORMULE", PAGE 108

Exercice 1 : Vous avez 2 possibilités pour le faire :

- Méthode la plus simple en débutant :
 - vous passez une première formule qui passe tout le prénom en minuscules :

```
[TECHNICIENS]Prénom:=Minusc([TECHNICIENS]Prénom)
```

- puis vous n'appliquez la majuscule qu'au 1er caractère :

```
[TECHNICIENS]Prénom[[1]]:=Majusc([TECHNICIENS]Prénom[[1]])
```

Vous remarquez l'utilisation des "indices de chaîne" qui permettent d'adresser un caractère particulier dans la chaîne.

Note : Sur Mac OS, vous pouvez utiliser le même indice de chaîne [[n]], il se convertira seul en $\leq n \geq$ (voir exemple suivant).

- Méthode un peu plus compliquée... mais vous verrez, ça vient très vite ! Vous écrivez une seule formule :

Windows :

```
[TECHNICIENS]Prénom:=Majusc([TECHNICIENS]Prénom[[1]])+Minusc(Sous chaîne([TECHNICIENS]Prénom;2))
```

Mac OS :

```
[TECHNICIENS]Prénom:=Majusc([TECHNICIENS]Prénom≤1≥)+Minusc(Sous chaîne([TECHNICIENS]Prénom;2))
```

Si vous préférez passer l'ensemble des paramètres à la fonction "Sous chaîne", vous pouvez écrire la ligne comme ceci :

```
[TECHNICIENS]Prénom:=Majusc([TECHNICIENS]Prénom[[1]])+Minusc(Sous chaîne([TECHNICIENS]Prénom;2;Longueur([TECHNICIENS]Prénom)-1))
```

EXERCICE : "HÉRITAGE DE FORMULAIRES", PAGE 114

- 1 Affichez un des formulaires "Entrée".
- 2 Sélectionnez les boutons et les textes associés puis coupez-les.
- 3 Affichez le formulaire HERITAGE_ENTREE.
- 4 Collez les boutons.
- 5 Revenez sur le formulaire ENTREE.
- 6 Modifiez les propriétés d'héritage (table + formulaire).

Vos boutons doivent réapparaître.

Sur les autres formulaires entrée, vous devez juste supprimer les boutons et paramétrer les propriétés d'héritage.

EXERCICE : "PROPRIÉTÉS DES OBJETS", PAGE 131

- *Réponse 1* : Puisque l'événement Sur chargement est coché, la méthode associée à l'objet est exécutée également au chargement du formulaire. Cet exemple vous permet de comprendre l'importance des événements ainsi que la solution de "test" que je vous propose à la ligne suivante.
- *Réponse 2* : Le fait d'avoir limité l'exécution de la recherche au seul événement Sur clic permet de maintenir l'événement Sur chargement coché tout en conservant un fonctionnement cohérent.

EXERCICE : "CALCULS ET FORMULES", PAGE 135

Bouton Chercher :

```

1 $Evt=Evenement formulaire
2 Au cas ou
3   | ($Evt=Sur clic)
4   |
5   | Au cas ou
6   |   | (Page formulaire courante=1)
7   |   | CHERCHER((TECHNICIENS))
8   |   | vNbEnreg:=Enregistrements trouves((TECHNICIENS))
9   |   |
10  |   | (Page formulaire courante=2)
11  |   | CHERCHER((INTERVENTIONS))
12  |   | vNbEnreg:=Enregistrements trouves((INTERVENTIONS))
13  |   |
14  |   | (Page formulaire courante=3)
15  |   | CHERCHER((MATERIELS))
16  |   | vNbEnreg:=Enregistrements trouves((MATERIELS))
17  |   |
18  |   | (Page formulaire courante=4)
19  |   | `PERIPHERIQUES la table n'est pas encore utilisée
20  |   |
21  |   | (Page formulaire courante=5)
22  |   | `RESEAUX la table n'est pas encore utilisée
23  |   |
24  |   | (Page formulaire courante=6)
25  |   | CHERCHER((LOGICIELS))
26  |   | vNbEnreg:=Enregistrements trouves((LOGICIELS))
27  |   |
28  |   | Fin de cas
29  |   |
30  |   |
31  |   | ($Evt=Sur chargement)
32  |   |
33  |   | Fin de cas

```

Bouton Toutes :

```

1 Au cas ou
2   | (Page formulaire courante=1)
3   | TOUT SELECTIONNER((TECHNICIENS))
4   | vNbEnreg:=Enregistrements trouves((TECHNICIENS))
5   |
6   | (Page formulaire courante=2)
7   | TOUT SELECTIONNER((INTERVENTIONS))
8   | vNbEnreg:=Enregistrements trouves((INTERVENTIONS))
9   |
10  | Fin de cas

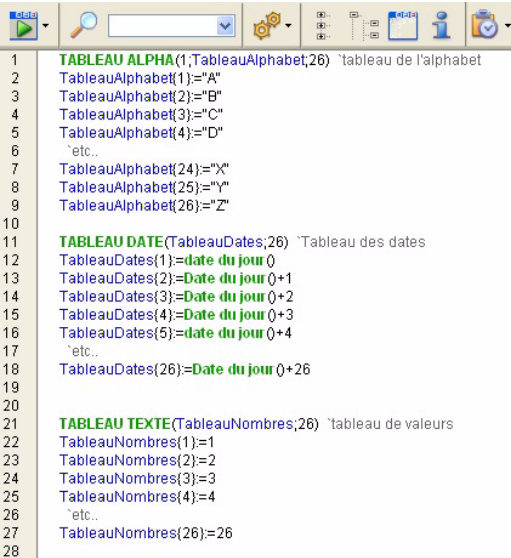
```



Rappel : pour afficher rapidement la méthode d'un objet (bouton, variable, champ...) il suffit d'appuyer sur la touche Alt et de cliquer sur l'objet. Si la méthode existe, elle s'ouvre, si elle n'existe pas 4D la crée et l'ouvre

EXERCICE : "PRÉSENTATION DES VARIABLES", PAGE 140.

Une première méthode consiste à définir les tableaux ligne à ligne comme ceci :

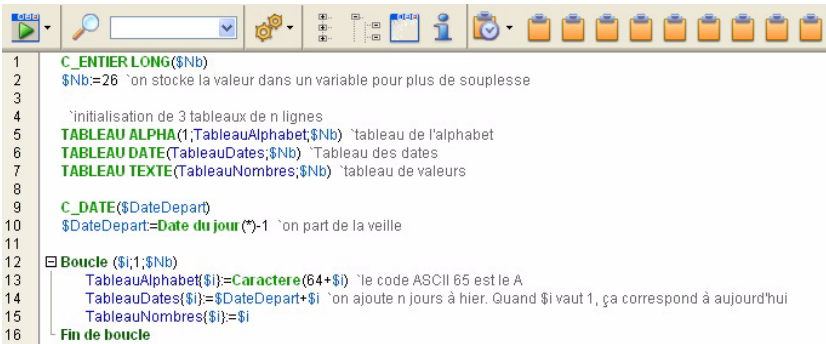


```

1  TABLEAU ALPHA(1,TableauAlphabet,26) `tableau de l'alphabet
2  TableauAlphabet(1):="A"
3  TableauAlphabet(2):="B"
4  TableauAlphabet(3):="C"
5  TableauAlphabet(4):="D"
6  `etc..
7  TableauAlphabet(24):="X"
8  TableauAlphabet(25):="Y"
9  TableauAlphabet(26):="Z"
10
11 TABLEAU DATE(TableauDates,26) `Tableau des dates
12 TableauDates(1):=date du jour 0
13 TableauDates(2):=Date du jour 0+1
14 TableauDates(3):=Date du jour 0+2
15 TableauDates(4):=Date du jour 0+3
16 TableauDates(5):=date du jour 0+4
17 `etc..
18 TableauDates(26):=Date du jour 0+26
19
20
21 TABLEAU TEXTE(TableauNombres,26) `tableau de valeurs
22 TableauNombres(1)=1
23 TableauNombres(2)=2
24 TableauNombres(3)=3
25 TableauNombres(4)=4
26 `etc..
27 TableauNombres(26)=26
28

```

Une méthode un peu plus optimisée consiste à utiliser une boucle pour remplir les tableaux :



```

1  C_ENTIER LONG($Nb)
2  $Nb:=26 `on stocke la valeur dans un variable pour plus de souplesse
3
4  `initialisation de 3 tableaux de n lignes
5  TABLEAU ALPHA(1,TableauAlphabet,$Nb) `tableau de l'alphabet
6  TABLEAU DATE(TableauDates,$Nb) `Tableau des dates
7  TABLEAU TEXTE(TableauNombres,$Nb) `tableau de valeurs
8
9  C_DATE($DateDepart)
10 $DateDepart:=Date du jour (*)-1 `on part de la veille
11
12 Boucle ($i,1,$Nb)
13   TableauAlphabet($i):=Caractere(64+$i) `le code ASCII 65 est le A
14   TableauDates($i):=$DateDepart+$i `on ajoute n jours à hier. Quand $i vaut 1, ça correspond à aujourd'hui
15   TableauNombres($i):=$i
16 Fin de boucle

```


EXERCICE : "PROGRAMMATION GÉNÉRIQUE (SANS POINTEURS)", PAGE 151

```

1  C_ENTIER LONG($?;$NumPage)
2  $NumPage:= $?
3
4  ALLER A PAGE($NumPage)
5


```

```

1  C_ENTIER LONG($?;$NumPage)
2  $NumPage:= $?
3
4  ALLER A PAGE($NumPage) `déplacement vers la bonne page
5
6  `En fonction de la page, on prévoit un traitement possible
7  Au cas ou
8  : ($NumPage=1)
9
10 : ($NumPage=2)
11
12 : ($NumPage=3)
13
14 : ($NumPage=4)
15
16 : ($NumPage=5)
17
18 : ($NumPage=6)
19
20 : ($NumPage=7)
21
22 : ($NumPage=8) `rechercher l'ensemble des fiches Enumérations qui contiennent les différents paramètres personnalisables de la base de données
23   TOUT SELECTIONNER([ENUMERATIONS]) `prendre toutes les fiches énumération
24   VALEURS DISTINCTES([ENUMERATIONS]Titre_Enum;Tab_Titres_Enums) `faire la liste des titres
25   Tab_Titres_Enums:=1 `se positionner sur le premier
26   CHERCHER([ENUMERATIONS];[ENUMERATIONS]Titre_Enum=Tab_Titres_Enums{Tab_Titres_Enums}) `rechercher les valeurs énum correspondantes
27   TRIER([ENUMERATIONS];[ENUMERATIONS]Ordre_Tri;>) `les trier par ordre croissant
28
29   Sinon
30
31 Fin de cas
32
33 `Désactivation des boutons de navigation en fonction de la page sur laquelle on se trouve.]
34 SI ($NumPage<=6) `si on se trouve sur une des 6 premières pages les boutons seront actifs
35   ACTIVER BOUTON(*;"bNav@")
36   Sinon
37   INACTIVER BOUTON(*;"bNav@")
38   Fin de si
39

```

EXERCICE : "POINTEURS", PAGE 157



```

1 C_TEXTE($1;$Action)
2 C_ENTIER LONG($2;$NumPage)
3
4 $Action:=$1
5 $NumPage:=$2
6
7 C_POINTEUR($PointeurTable) `pointeur vers la table correspondant à la page affichée
8
9 `on indique ci-dessous "cette" table
10 ▣ Au cas ou
11   ▣ : ($NumPage=1) `Techniciens
12     | $PointeurTable:=->[TECHNICIENS]
13   ▣ : ($NumPage=2) `Interventions
14     | $PointeurTable:=->[INTERVENTIONS]
15   ▣ : ($NumPage=3) `Matériels
16     | $PointeurTable:=->[MATERIELS]
17   ▣ : ($NumPage=4) `Périphériques
18     | $PointeurTable:=->[PERIPHERIQUES]
19   ▣ : ($NumPage=5) `Réseau
20     | $PointeurTable:=->[RESEAUX]
21   ▣ : ($NumPage=6) `Logiciels
22     | $PointeurTable:=->[LOGICIELS]
23
24   `les lignes ci-dessous sont en commentaires car les boutons en sont pas actifs lorsqu'elles sont affichées.
25   `: ($NumPage=7) `Systèmes
26     | $PointeurTable:=->[SYSTEMES]
27   `: ($NumPage=8) `Consommables
28     | $PointeurTable:=->[Enumerations]
29   ▣ Sinon
30     | `cas non prévu actuellement
31 - Fin de cas

```

(la suite page suivante)

```

32
33   `a ce stade, on connait la table concernée et on va "dépointer la variable" pour connaître la table "que tu m'as montrée"
34   Au cas ou
35     ☐: ($Action="Terminer") `Terminer
36         NE PAS VALIDER `action valable quelle que soit la page sélectionnée
37     ☐: ($Action="Ajouter") `Ajouter
38         AJOUTER ENREGISTREMENT($PointeurTable->,)
39     ☐: ($Action="Toutes") `Toutes
40         TOUT SELECTIONNER($PointeurTable->)
41         vNbEnreg:=Enregistrements trouvés($PointeurTable->)
42     ☐: ($Action="Chercher") `Chercher
43         CHERCHER($PointeurTable->)
44         vNbEnreg:=Enregistrements trouvés($PointeurTable->)
45     ☐: ($Action="Sélection") `Sélection
46         $NomEnsemble:="Userset_"+Chaîne(Nombre de millisecondes)
47         LIRE ENREGISTREMENTS MARQUÉS($PointeurTable->,$NomEnsemble)
48         UTILISER ENSEMBLE($NomEnsemble)
49         vNbEnreg:=Enregistrements trouvés($PointeurTable->)
50     ☐: ($Action="Trier") `Trier
51         TRIER($PointeurTable->)
52     ☐: ($Action="Imprimer") `Imprimer
53         IMPRIMER SELECTION($PointeurTable->)
54     ☐: ($Action="Etats") `Etats
55         OR ETAT($PointeurTable->,Caractere(1),Vrai,Vrai,Vrai)
56     ☐: ($Action="Etiquettes") `Etiquettes
57         IMPRIMER ETIQUETTES($PointeurTable->,Caractere(1))
58     ☐: ($Action="Supprimer") `Supprimer
59         $NomEnsemble:="Userset_"+Chaîne(Nombre de millisecondes)
60         LIRE ENREGISTREMENTS MARQUÉS($PointeurTable->,$NomEnsemble)
61         $Nb:=Enregistrements dans ensemble($NomEnsemble) `indique le nombre d'enregistrements sélectionnés par l'utilis-
62     Au cas ou
63         ☐: ($Nb=0) `pas d'enregistrement
64             ALERTE("Il faut sélectionner au moins 1 enregistrement à supprimer")
65         ☐ Sinon
66             CONFIRMER(Chaîne($Nb)+" enregistrement(s) à supprimer ?","Supprimer","Annuler")
67             ☐ Si (OK=1)
68                 UTILISER ENSEMBLE($NomEnsemble)
69                 SUPPRIMER SELECTION($PointeurTable->)
70             ☐ Sinon
71                 `on ne fait rien
72             Fin de si
73         Fin de cas
74         EFFACER ENSEMBLE($NomEnsemble)
75         vNbEnreg:=Enregistrements trouvés($PointeurTable->)
76     Fin de cas

```



ASTUCE

Programmation avancée : Pour éviter de définir la table concernée à chaque fois que nous appelons la méthode, nous pouvons utiliser un tableau de pointeurs qui contiendra à la première ligne un pointeur vers la table [techniciens], en ligne 2 un pointeur vers la table [interventions], etc.

```

1  TABLEAU POINTEUR(◊TabPointeurTables;10)  *tableau contenant des pointeurs vers les tables
2
3  ◊TabPointeurTables(1) :=->[TECHNICIENS]
4  ◊TabPointeurTables(2) :=->[INTERVENTIONS]
5  ◊TabPointeurTables(3) :=->[MATERIELS]
6  ◊TabPointeurTables(4) :=->[PERIPHERIQUES]
7  ◊TabPointeurTables(5) :=->[RESE AUX]
8  ◊TabPointeurTables(6) :=->[LOGICIELS]
9  ◊TabPointeurTables(7) :=->[SYSTEMES]
10 ◊TabPointeurTables(8) :=->[ENUMERATIONS]

```

Dans ce cas, on tient compte du numéro de page pour adresser les lignes du tableau. Chaque ligne du tableau (◊TabPointeurTables{\$NumPage}) est un pointeur qu'il faut dépointer (->) :

```

34  Au cas ou
35  : ($Action="Terminer")  *Terminer
36  NE PAS VALIDER  *action valable quelle que soit la page sélectionnée
37  : ($Action="Ajouter")  *Ajouter
38  AJOUTER ENREGISTREMENT(◊TabPointeurTables{$NumPage}->)*
39  : ($Action="Toutes")  *Toutes
40  TOUT SELECTIONNER(◊TabPointeurTables{$NumPage}->)
41  : ($Action="Chercher")  *Chercher
42  CHERCHER(◊TabPointeurTables{$NumPage}->)

```

EXERCICE : "ÉVÉNEMENTS", PAGE 163

Vous pouvez également l'écrire de la manière optimisée en utilisant le pointeur **Self** qui désigne l'objet sur lequel on agit.

```

1  $evt:=Evenement formulaire
2  Au cas ou
3  : ($evt=Sur donnees modifiees)  *lorsqu'on modifie la règle
4  [INTERVENTIONS]Avancement:=Self->  *affecter la valeur de la variable au champ
5
6  : ($evt=Sur chargement)  *quand on ouvre une fiche Intervention
7  C_ENTIER LONG(TH_Avancement)  *taper la variable TH_Avancement
8  Self->:=[INTERVENTIONS]Avancement  *affecter la valeur du champ à la variable
9  Fin de cas
10

```

Vous remarquez qu'on dépointe **Self** car c'est un pointeur. On écrit donc **Self->** quand on veut adresser l'objet pointé.



ASTUCE

Il est important de prendre en compte l'ordre d'exécution des méthodes lors de vos débogages. Si vous avez mis du code dans l'événement Sur chargement de votre méthode formulaire ET de certains de vos objets, ce sont d'abord les méthodes objets qui sont exécutées puis la méthode formulaire.

Les méthodes objets sont exécutées dans l'ordre de saisie (qui correspond également à l'ordre de profondeur de plan des objets).

EXERCICE : "TABLEAUX, POP UP, LISTBOX", PAGE 173

Programmation identique au tableau des heures de début.



ASTUCE

Pour des questions d'optimisation (éviter de recréer le tableau à chaque ouverture d'enregistrement), vous pouvez créer le tableau une seule fois à l'ouverture de la base en appelant la méthode INIT_TABLEAUX (à créer). Dans cette méthode vous créez un tableau interprocess \diamond Tabheures puis recopiez le contenu dans votre tableau sur chargement comme indiqué ci-dessous.

```

Méthode : INIT_TABLEAUX
1 C_ENTIER LONG($TailleTableauHeures)
2 $TailleTableauHeures:=25
3 TABLEAU ENTIER LONG( $\diamond$ Tabheures;$TailleTableauHeures)
4 C_HEURE($Heuredeb)
5 $Heuredeb:='07:30:00'
6 Boucle ($i,1;$TailleTableauHeures)
7    $\diamond$ Tabheures($i):=$Heuredeb+'00:30:00'*$i
8 Fin de boucle
9

```

```

1 $evt:=Evenement formulaire
2 Au cas ou
3   : ($evt=Sur_chargement) `avant l'affichage du formulaire
4     TABLEAU ENTIER LONG(TabHeuresFin,0) `réservé un espace en mémoire
5     COPIER TABLEAU( $\diamond$ Tabheures;Self->) `copier le tableau créé au démarrage de la base
6
7   : ($evt=Sur_clic) `lors du clic dans le popup
8     Si (Self->#0) `si une ligne est bien sélectionnée
9       [INTERVENTIONS]Heure_Fin:=Self->{Self->} `récupérer la valeur choisie dans le champ
10    Fin de si
11 Fin de cas
12

```

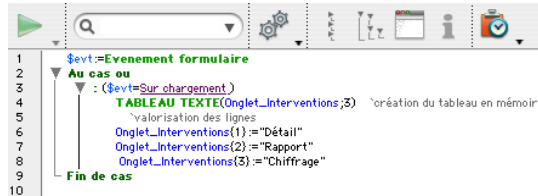
EXERCICE : "4D WRITE", PAGE 179

Vous avez :

- quitté 4D,

- ajouté le plugin 4D View dans le dossier PlugIns,
- redémarré votre base de données,
- ajouté une page à votre formulaire.

Bien sûr, vous n'avez pas oublié d'ajouter une ligne au tableau qui gère l'onglet afin de pouvoir accéder à cette troisième page :



```

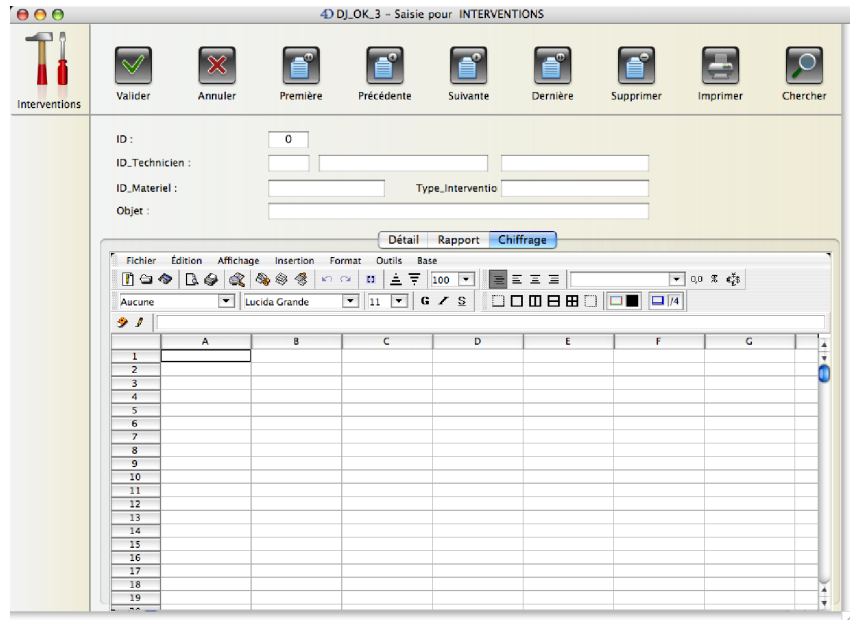
1 $evt:=Evenement formulaire
2 Au cas ou
3 : ($evt=Sur_chargement)
4   TABLEAU TEXTE(Onglet_Interventions;3) 'création du tableau en mémoire
5   ^valorisation des lignes
6   Onglet_Interventions(1):="Détail"
7   Onglet_Interventions(2):="Rapport"
8   Onglet_Interventions(3):="Chiffrage"
9   Fin de cas
10

```

Ensuite, vous avez ajouté un champ Chiffrage_

INTERVENTIONS	
ID	2 ²⁴
ID_Technicien	A
Objet	A
Descriptif	T
Date_Debut	11
Heure_Debut	11
Date_Fin	11
Heure_Fin	11
Commentaire	T
Avancement	2 ¹⁴
Type_Intervention	A
ID_Materiel	A
Rapport_	11
Chiffrage_	11

Enfin, vous avez créé la zone de plug-in dans le formulaire, indiqué son type (4D View) et donné le nom "Chiffrage" afin d'obtenir en saisie :



ASTUCE

Pour travailler en plein écran dans avec vos plug-ins, demander "Aller en pleine page" du menu Fichier. Pour revenir de ce mode pleine page, sélectionnez la même commande qui est renommée "Retour au formulaire".

EXERCICE : "FENÊTRE ET NAVIGATION", PAGE 186

Méthode ALERTE_DIALOGUE :

```

1 C_TEXTE($1;$2;$3)
2 C_TEXTE(vTitrefenetreAlerte,vMessageAlerte,vMessageAlerteComplement)
3   *valoriser les variables pour faciliter la saisie
4 vTitrefenetreAlerte=$1
5 vMessageAlerte=$2
6 vMessageAlerteComplement=$3
7
8   *créer la fenêtre (conteneur) dans laquelle on affiche le dialogue (contenu)
9 $Fenetre=Creer fenetre formulaire("MESSAGE";Fenetre standard ;Centrée horizontalement ;Centrée verticalement)
10  *changer le titre de la fenêtre
11 CHANGER TITRE FENETRE(vTitrefenetreAlerte)
12  *afficher le dialogue avec les 2 variables qui ont reçu la valeur des paramètres $2 et $3
13 DIALOGUE("MESSAGE")
14  *une fois le dialogue refermé, refermer la fenêtre
15 FERMER FENETRE($Fenetre)
16

```

Appel de la méthode à partir d'une autre méthode (objet, projet...) :

```

1 C_TEXTE($Titre;$Message;$Complement)
2 $Titre:="Erreur de saisie"
3 $Message:="Vous ne pouvez entrer une valeur texte dans ce champ"
4 $Complement:="Pour plus d'informations, reportez-vous à la documentation fournie avec le logiciel."
5 ALERTE_DIALOGUE ($Titre;$Message;$Complement)
6

```



Nous avons laissé les textes saisissables, ce qui permet à l'utilisateur de copier-coller le message plutôt que de le recopier à la main s'il doit nous en informer.

EXERCICE : "4D INTERNET COMMANDS", PAGE 192

Vous devez ajouter un bouton sur le formulaire projet MESSAGE (le formulaire qui affiche les messages d'alerte) :

Paramétrez la méthode du bouton pour qu'il prenne en compte les informations du message et les envoie à une adresse que vous avez définie :

```

1 $EmailDestinaire:="support@4d.fr"
2 $EmailEmetteur:="utilisateur@4d.fr"
3 $Objet:="Message d'erreur sur : "+vTitrefenetreAlerte
4 $Message:=vMessageAlerte+(Caractere(13)*2)+vMessageAlerteComplement
5 $EmailCC:="utilisateur@4d.fr"
6 EMAIL_ENVOI_SIMPLIFIE ($EmailDestinaire;$EmailEmetteur;$Objet;$Message;$EmailCC)
7
8

```


EXERCICE : "MOTS DE PASSE ET GROUPES", PAGE 196

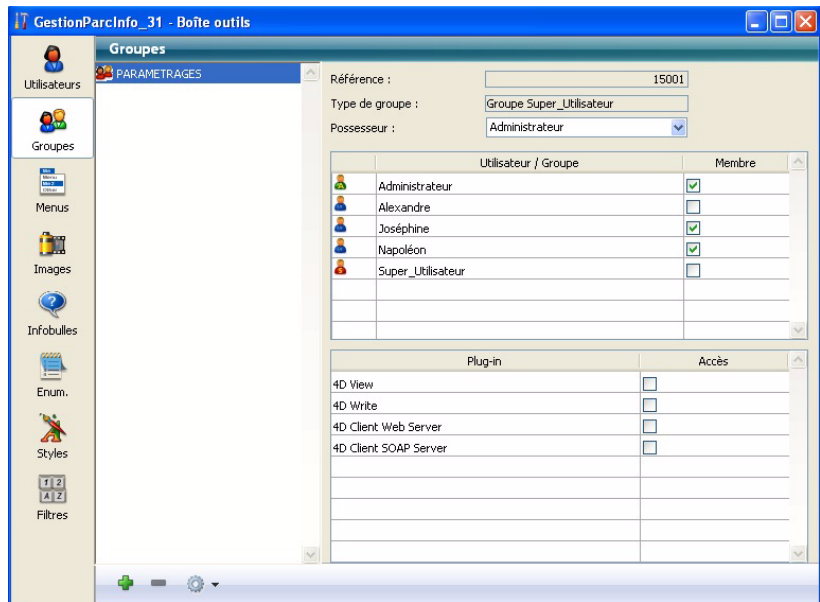
Voici une manière de programmer une limitation d'accès :

```

33
34 `Désactivation des boutons de navigation en fonction de la page sur laquelle on se trouve.
35 Si ($Numpage<=6) `si on se trouve sur une des 6 premières pages les boutons seront actifs
36 ACTIVER BOUTON("bNav@") `on adresse les objets par leur nom d'objet, ce qui permet avec @ de les gérer par groupe
37 Sinon `on se situe sur la page 7 ou 8
38 Si (Appartient au groupe(Utilisateur courant,"PARAMETRAGES")) `si l'utilisateur connecté appartient au groupe qui a le droit
39 INACTIVER BOUTON("bNav@") `on inactive les boutons de navigation qui ne doivent pas être actifs dans ce contexte
40 Sinon `l'utilisateur n'a pas le droit d'accéder
41 ALERTE("Vous ne pouvez accéder à ces paramètres.") `message d'alerte
42 ALLER A PAGE(1) `renvoi forcé sur la page 1
43 Fin de si
44 Fin de si

```

Dans la base exemple j'ai créé 3 utilisateurs comme le montre l'image ci-dessous et n'en ai attribué que 2 au groupe PARAMETRAGE :

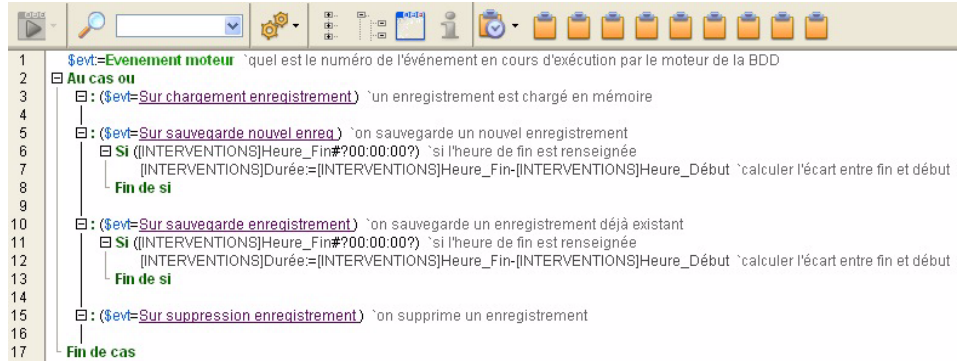


J'ai également ajouté le mot de passe "A" au Super_Utilisateur pour que 4D affiche la fenêtre des utilisateurs au démarrage.

Dans les bases exemple suivantes, j'ai supprimé le mot de passe.

EXERCICE : "TRIGGERS", PAGE 201

La même méthode doit s'appliquer lorsqu'on modifie un enregistrement ou lorsqu'on en crée un puisqu'on peut saisir une intervention passée pour laquelle on connaît l'heure de début et de fin.



```

1  $evt=Evenement moteur `quel est le numéro de l'événement en cours d'exécution par le moteur de la BDD
2  Au cas ou
3  |  | ($evt=Sur chargement enregistrement) `un enregistrement est chargé en mémoire
4  |  |
5  |  | ($evt=Sur sauvegarde nouvel enreg) `on sauvegarde un nouvel enregistrement
6  |  | | Si ([INTERVENTIONS]Heure_Fin#?00:00:00?) `si l'heure de fin est renseignée
7  |  | | | [INTERVENTIONS]Durée:=|[INTERVENTIONS]Heure_Fin-[INTERVENTIONS]Heure_Début `calculer l'écart entre fin et début
8  |  | | | Fin de si
9  |  |
10 |  | ($evt=Sur sauvegarde enregistrement) `on sauvegarde un enregistrement déjà existant
11 |  | | Si ([INTERVENTIONS]Heure_Fin#?00:00:00?) `si l'heure de fin est renseignée
12 |  | | | [INTERVENTIONS]Durée:=|[INTERVENTIONS]Heure_Fin-[INTERVENTIONS]Heure_Début `calculer l'écart entre fin et début
13 |  | | | Fin de si
14 |  |
15 |  | ($evt=Sur suppression enregistrement) `on supprime un enregistrement
16 |  |
17 |  | Fin de cas

```

Pour que la programmation soit prise en compte, j'ai coché les événements moteurs comme ceci :



J'ai également mis des points d'arrêt pour suivre ce qui se passe. Le champ Durée a été ajouté au formulaire entrée pour en vérifier le bon fonctionnement.

On peut optimiser ce trigger en créant une méthode *INTERVENTION_GESTION* à laquelle on passe le paramètre *Durée* :

```

1 C_TEXTE($1,$Action)
2 $Action=$1
3
4 Au cas ou
5   ($Action="Duree")
6   Si ([INTERVENTIONS]Heure_Fin#?00:00:00?) `si l'heure de fin est renseignée
7     [INTERVENTIONS]Durée=[INTERVENTIONS]Heure_Fin-[INTERVENTIONS]Heure_Début `calculer l'écart entre fin et début
8     Fin de si
9
10  ($Action="")
11  Sinon
12
13 Fin de cas

```

Ensuite, on modifie le trigger comme suit :

```

1 $evt=Evenement moteur `quel est le numéro de l'événement en cours d'exécution par le moteur de la BDD
2 Au cas ou
3   ($evt=Sur chargement enregistrement) `un enregistrement est chargé en mémoire
4   ($evt=Sur sauvegarde nouvel enreg.) `on sauvegarde un nouvel enregistrement
5   INTERVENTION_GESTION ("Duree")
6   ($evt=Sur sauvegarde enregistrement) `on sauvegarde un enregistrement déjà existant
7   INTERVENTION_GESTION ("Duree")
8   ($evt=Sur suppression enregistrement) `on supprime un enregistrement
9
10 Fin de cas

```

L'avantage est de n'avoir qu'une seule version de calcul, donc cohérente pour la base et de pouvoir appeler cette méthode à partir d'autres méthodes dans la base de données.



Pour ouvrir le trigger d'une table, vous pouvez utiliser le raccourci suivant dans la fenêtre de Structure : Alt+Double clic sur le nom de la table

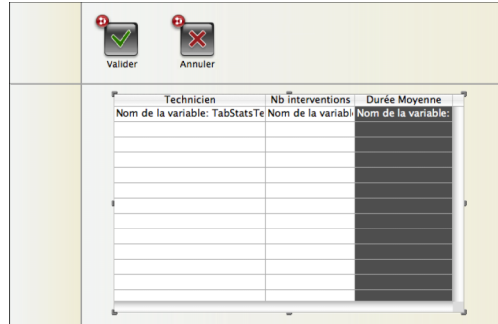
EXERCICE : "SÉLECTION COURANTE", PAGE 210

Dans la base exemple, j'ai ajouté un bouton qui appelle le calcul statistique et permet de l'exécuter directement à partir du formulaire. J'ai modifié la méthode STATISTIQUES de manière à ce qu'elle initialise les tableaux et affiche le formulaire et j'ai déplacé le reste du code dans la méthode objet du bouton. Vous commencez à bien naviguer dans les méthodes et formulaires alors... découvrez comment j'ai mis en place l'ensemble, ça vous aidera à concevoir vos solutions et à comprendre comment d'autres ont pu organiser leur programmation.

Note 1 : Le 3e tableau que vous créez est bien de type "Entier long". Les tableaux "Heure" n'existent pas dans 4D. Il utilise à la place les tableaux entier long. Pensez à l'indiquer dans les propriétés de la 3e colonne de votre Listbox.

Note 2 : pour que les moyennes de durées soient calculées, il faut bien évidemment que les durées soient renseignées... à vous de jouer, et n'oubliez pas les triggers !

Ajout de la colonne dans la Listbox :



Méthode du bouton :

```

1  C_DATE($DateDebut;$DateFin)
2  *Constitutions une date par rapport à l'année en cours
3  $DateDebut :=Date("01/01/"+Chaine(Annee de(Date du jour(*))) *on récupère l'année de la date du jour qu'on convertit en texte
4  $DateFin :=Ajouter a date($DateDebut;1;0;-1) *on ajoute 1 an et on retranche 1 journée. on obtient donc le 31/12 de l'année précédente
5
6  TOUJOURS SELECTIONNER([TECHNICIENS]) *créer une sélection courante avec l'ensemble des enregistrements
7
8  $Nb :=Enregistrements trouvés([TECHNICIENS]) *Combien y a-t'il d'enregistrements dans la sélection
9
10 TABLEAU ENTIER LONG(TabStatsIntervention;$Nb) *créer un tableau avec autant de lignes que de techniciens
11 TABLEAU TEXTE(TabStatsTechniciens;$Nb) *créer un tableau avec autant de lignes que de techniciens
12 TABLEAU ENTIER LONG(TabStatsDureeMoyennes;$Nb)
13
14 Tant que (Non(Fin de selection([TECHNICIENS]))) *pour chaque fiche Technicien
15     *effectuer un triement
16     CHERCHER([INTERVENTIONS];[INTERVENTIONS]ID_Technicien=[TECHNICIENS]ID) *retrouve les interventions du technicien
17
18     *On cherche maintenant avec 2 critères dans les enregistrements trouvés lors de la recherche précédente
19     CHERCHER DANS SELECTION([INTERVENTIONS];[INTERVENTIONS]Date_Debut=[$DateDebut];*) * interventions supérieures ou égales à la borne inférieure
20     CHERCHER DANS SELECTION([INTERVENTIONS]; & ;[INTERVENTIONS]Date_Debut<=[$DateFin]) *et inférieures ou égales à la borne supérieure
21
22     CHERCHER DANS SELECTION([INTERVENTIONS];[INTERVENTIONS]Heure_Fin#100.00.00) *on ne conserve que les interventions terminées
23
24     *A ce stade, la sélection courante est modifiée et correspond à nos souhaits.
25
26     $Indice :=Numero dans selection([TECHNICIENS]) *indice précisant dans quelle ligne du tableau on doit stocker le résultat
27     TabStatsIntervention($Indice) :=Enregistrements trouvés([INTERVENTIONS])
28     TabStatsTechniciens($Indice) :=[TECHNICIENS]Nom
29     TabStatsDureeMoyennes($Indice) :=Moyenne([INTERVENTIONS]Duree)
30
31     ENREGISTREMENT SUIVANT([TECHNICIENS]) *aller à la fiche suivante
32 Fin tant que
33
34
35 *créer la fenêtre (conteneur) dans laquelle on affiche le dialogue (contenu)
36 $Fenetre :=Créer fenetre formulaire("STATISTIQUES" ;Fenetre_standard ;Centrée horizontalement ;Centrée verticalement)
37 DIALOGUE("STATISTIQUES") *afficher le dialogue
38 FERMER FENETRE($Fenetre) *une fois le dialogue refermé, refermer la fenêtre

```



Pour combiner plusieurs recherches, il suffit de terminer chaque ligne (sauf la dernière) par les 2 caractères ;* et de lier les recherches par les conjonctions de coordination :

- & (et)
- | (ou) (il s'agit du Pipe = trait vertical obtenu par AltGr+6 sur Windows et Alt+Maj+L sur Mac OS.
- # (sauf)

EXERCICE : "ENSEMBLES ET SÉLECTIONS TEMPORAIRES", PAGE 216

Dans la méthode `NAVIGATION_FONCTIONS`, vous modifiez les lignes relatives à l'action "Ajouter" de la manière suivante :

```

: ($Action="Ajouter") `Ajouter
`Rappel : un nom d'ensemble est associé à une table, on ne peut donc pas avoir le même nom d'ensemble associé à plusieurs tables
$NomEnsembleInitial=$Chaine(Nombre de millisecondes) `on constitue le nom de l'ensemble pour qu'il soit unique. Le nom n'a pas besoin d'être explicite.
NOMMER ENSEMBLE($PointeurTable->,$NomEnsembleInitial) `on conserve la liste des enregistrements actuellement dans la sélection
$NomEnsembleAjout=$Chaine(Nombre de ticks) `nom du 2è ensemble qui contiendra les ajouts de fiches
ENSEMBLE VIDE($PointeurTable->,$NomEnsembleAjout) `création d'un 2è ensemble (vide celui-ci)
Repeter `permettre l'ajout de plusieurs fiches d'affiliée
  AJOUTER ENREGISTREMENT($PointeurTable->,*;*) `ajout d'un nouvel enregistrement
  Si (OK=1) `si l'utilisateur a validé la fiche
    ADJOINDRE ELEMENT($PointeurTable->,$NomEnsembleAjout) `on ajoute cette nouvelle fiche à l'ensemble des ajouts
  Sinon `la fiche n'a pas été validée
    `on ne fait rien au niveau de l'ensemble
  Fin de si
Jusque (OK=0) `quand l'utilisateur veut arrêter, il clique sur Annuler => la variable OK prend la valeur 0
`à ce stade, on dispose de 2 ensembles
REUNION($NomEnsembleInitial,$NomEnsembleAjout,$NomEnsembleInitial) `on cumule les fiches de l'ensemble des ajouts à l'ensemble de départ.
`Les fiches sont cumulées en mémoire dans l'ensemble mais ne deviennent pas pour autant la sélection courante
UTILISER ENSEMBLE($NomEnsembleInitial) `on demande à ce que les fiches indiquées dans l'ensemble deviennent les fiches de la sélection courante

```

... et puisque vous avez décidé de programmer "proprement", nettoyez la mémoire une fois que nous n'avons plus besoin des ensembles :

```

`Libérons la mémoire occupée par les ensembles
EFFACER ENSEMBLE($NomEnsembleInitial)
EFFACER ENSEMBLE($NomEnsembleAjout)

```

EXERCICE : "PROCESS", PAGE 222

1 Créez le formulaire suivant :

Début compteur	vDebutCompt
Il est	vHeure
Temps écoulé	vTempsEcoule
Fermer le compteur	

2 Modifiez la méthode formulaire comme ceci :

```

1  $Evt:=Evenement formulaire `récupérer l'événement en cours de réalisation
2  Au cas ou
3  : ($Evt=Sur chargement) `avant l'affichage du formulaire
4    C_HEURE(vheure;vDebutCompteur;vTempsEcoule) `initialiser les variables
5    vDebutCompteur:=Heure courante `indiquer l'heure de départ
6    FIXER MINUTEUR(60) `demander au formulaire de générer l'événement sur minuteur toutes les secondes
7    `exprimé en Ticks (1tick = 1/60è de seconde)
8
9  : ($Evt=Sur minuteur) `chaque fois que l'événement est généré
10   vHeure:=Heure courante `mettre l'heure à jour
11   vTempsEcoule:=vHeure-vDebutCompteur `calculer le temps passé
12
13  : ($Evt=Sur libération)
14   FIXER MINUTEUR(0) `arrête la génération de l'événement sur minuteur
15
16  Fin de cas
17

```

3 Dans les propriétés du formulaire, cochez les événements que vous utilisez dans la méthode.

4 Ecrivez la méthode de création de process :

```

1  `lors de l'appel simple de la méthode elle ne reçoit pas de paramètre
2  Si (Nombre de parametres=0) `si aucun paramètre n'est reçu
3  `nous créons le process
4  C_ENTIER LONG(◇ProcessCompteur) `variable contenant le numéro du process
5  ◇ProcessCompteur:=Nouveau process(Nom methode courante;1024*1024;"Compteur temps";"Paramètre fictif";*)
6
7  Sinon `si la méthode reçoit au moins un paramètre
8
9  `créer la fenêtre (conteneur) dans laquelle on affiche le dialogue (contenu)
10 $Fenetre:=Creer fenetre formulaire("COMPTEUR_TEMPS";Fenetre standard ;Centrée horizontalement ;Centrée verticalement)
11 DIALOGUE("COMPTEUR_TEMPS") `afficher le dialogue
12 FERMER FENETRE($Fenetre) `une fois le dialogue refermé, refermer la fenêtre
13
14 Fin de si
15

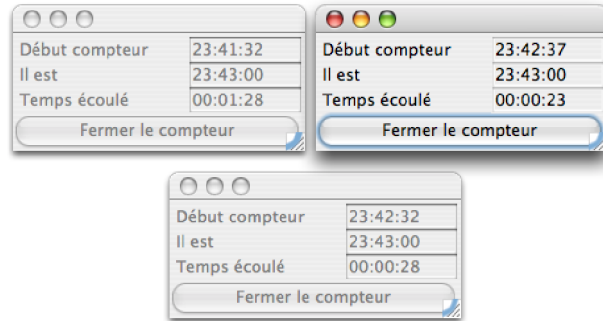
```

En exécutant cette méthode, vous devez obtenir le compteur temps à l'écran et pouvez continuer à travailler sur autre chose.



ASTUCE

L'étoile à la fin de la commande `NOUVEAU PROCESS` permet de ne pas recréer un même process si un process de même nom est déjà en cours d'exécution. Si vous souhaitez démarrer plusieurs compteurs temps, enlevez le `;` et exécutez plusieurs fois la méthode.*




Vous pourrez ensuite mettre en place un bouton qui permet de stopper le calcul du temps écoulé, ajouter un texte saisissable pour indiquer à quoi correspond le temps passé et par exemple à la fermeture du compteur, enregistrer le temps passé et le motif dans une table. Vous aurez ainsi une trace des temps passés.

Pour une meilleure ergonomie, tous les compteurs temps démarrés pourraient s'afficher dans une seule et même listbox mise à jour par une communication interprocess (via les commandes `LIRE VARIABLE PROCESS` ou `ECRIRE VARIABLE PROCESS`).

Pour démarrer votre compteur temps, vous pouvez utiliser un bouton, une ligne de menu, un démarrage automatique à l'ouverture d'un formulaire, etc.

EXERCICE : "SQL", PAGE 229

Exercice 1 :



```

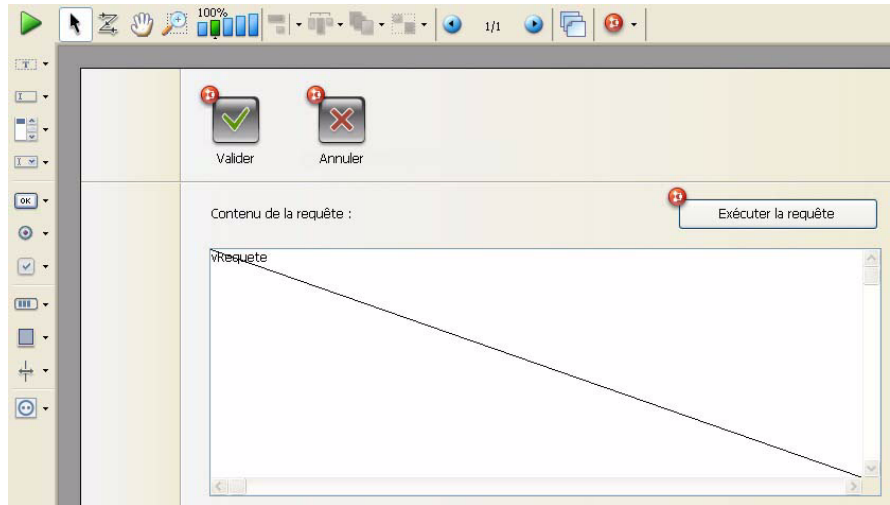
1 C_TEXTE($Requete) `variable qui contient la requête SQL
2 C_ENTIER LONG($NbFiches) `variable recevant le nombre de fiches suite à l'exécution de la requête SQL
3
4 $NbTables:=Lire numero derniere table `Plus grand numéro de table
5
6 TABLEAU TEXTE(Tab_Nom_Tables;0) `tableau devant contenir les noms de tables
7 TABLEAU ENTIER LONG(Tab_Nb_Fiches;0) `tableau devant contenir le nombre d'enregistrements de chaque table
8
9 [Boucle ($i;1;$NbTables) `pour tous les numéros de tables
10   [Si (Est un numero de table valide($i)) `si c'est une table existante
11     $NomTable:=Nom de la table(Table($i)) `permet de récupérer le nom de la table
12
13     $Requete:="SELECT COUNT(*)" `on veut compter le nombre de fiches
14     $Requete:=$Requete+"FROM "+$NomTable+" " `on ajoute le nom de la table et un espace après la commande FROM
15     $Requete:=$Requete+"INTO :$NbFiches" `on indique dans quelle variable le moteur SQL devra mentionner le résultat
16
17     [Debut SQL `exécution de la commande préparée ci-dessus
18       EXECUTE IMMEDIATE :$Requete;
19     ]
20     ]
21     AJOUTER A TABLEAU(Tab_Nom_Tables;$NomTable) `on ajoute le nom de la table
22     AJOUTER A TABLEAU(Tab_Nb_Fiches;$NbFiches) `ajout du nom de la table
23
24   ]
25 ]
26

```

Une fois les 2 tableaux remplis, vous pouvez les intégrer à un formulaire et les présenter dans un dialogue.

Vous pouvez par exemple intégrer ces tableaux dans votre formulaire navigation sur une page existante (Préférences par exemple) ou créer une nouvelle page "Statistiques" qui regroupera l'ensemble des statistiques que vous avez créées.

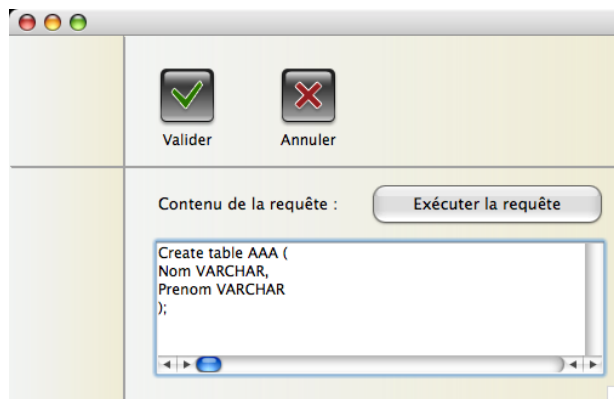
Exercice 2 : Créez un formulaire comme celui-ci :



Pour le créer rapidement, j'ai profité de l'héritage du formulaire *HERITAGE_DIALOGUES*. La méthode du bouton "Exécuter la requête" se présente sous la forme :



A l'exécution, vous pouvez écrire votre requête SQL dans le formulaire et l'exécuter :



Une fois ces principes compris, vous pourrez améliorer le système en mettant à disposition de vos utilisateurs la liste des tables, la liste des champs, des commandes SQL, etc.

EXERCICE : "BLOBS", PAGE 235

La méthode *BLOB_PREFERENCES* peut s'écrire de la manière suivante :

```

1  C_TEXTE($F1;$Action;$NomDocumentPreferences)
2  C_BLOB(vBlobPreferences) `variable interprocess car plusieurs appels successifs à la méthode
3  C_ENTIER LONG($PositionDansBlob)
4
5  $Action:=F1
6
7  $NomDocumentPreferences:="Preferences.4DZ" `nom que l'on souhaite donner au fichier de préférences
8
9  ▢ Au cas ou
10  ▢ ($Action="DeblobLesVariables") `extrait les variables du blob
11  $PositionDansBlob:=0
12  BLOB VERS VARIABLE(vBlobPreferences;<=>EcranDemarrage;$PositionDansBlob)
13  BLOB VERS VARIABLE(vBlobPreferences;<=>CouleurFond;$PositionDansBlob)
14  BLOB VERS VARIABLE(vBlobPreferences;<=>PoliceDefault;$PositionDansBlob)
15  BLOB VERS VARIABLE(vBlobPreferences;<=>TaillePoliceDefault;$PositionDansBlob)
16
17  ▢ ($Action="BlobeLesVariables") `Stocke les variables dans le blob
18  FIXER TAILLE BLOB(vBlobPreferences;0) `on vide le blob
19  VARIABLE VERS BLOB(<=>EcranDemarrage;vBlobPreferences;")
20  VARIABLE VERS BLOB(<=>CouleurFond;vBlobPreferences;")
21  VARIABLE VERS BLOB(<=>PoliceDefault;vBlobPreferences;")
22  VARIABLE VERS BLOB(<=>TaillePoliceDefault;vBlobPreferences;")
23
24  ▢ ($Action="LectureDisque") `Lit le blob sur le disque
25  ▢ Si (tester chemin acces($NomDocumentPreferences)=Est un document) `si le document de préférences existe
26  DOCUMENT VERS BLOB($NomDocumentPreferences;vBlobPreferences) `charge le document dont le nom est indiqué
27  ▢ Sinon `le document de préférences n'existe pas, nous le créerons en quittant la base
28
29  Fin de si
30
31  ▢ ($Action="EcritureDisque") `Ecrit le blob sur le disque
32  BLOB VERS DOCUMENT($NomDocumentPreferences;vBlobPreferences)
33
34  ▢ ($Action="InitVariables") `Initialise les variables. Normalement elles sont ensuite valorisées en fonction des choix de l'utilisateur
35  C_TEXTE(<=>EcranDemarrage)
36  C_ENTIER LONG(<=>CouleurFond)
37  C_TEXTE(<=>PoliceDefault)
38  C_ENTIER LONG(<=>TaillePoliceDefault)
39
40  ▢ Sinon
41  ALERTE("Ce cas n'est pas prévu.")
42  Fin de cas

```

L'appel de la méthode se fait à l'ouverture, dans la méthode *INITIALISATIONS*, comme ceci :

```

1  INIT_TABLEAUX
2
3
4  BLOB_PREFERENCES ("InitVariables")
5  BLOB_PREFERENCES ("LectureDisque")
6  BLOB_PREFERENCES ("DeblobLesVariables")
7
8  $NumFenetre:=Creer fenetre formulaire("Navigation")
9  DIALOGUE("Navigation")
10 FERMER FENETRE($NumFenetre)

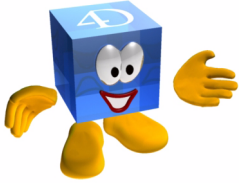
```

Et à la fermeture, comme cela (j'ai utilisé la méthode *Message_Fermeture* appelée dans la méthode base *Sur Fermeture*) :

```
1  ● BLOB_PREFERENCES ("BlobeLesVariables")
2  BLOB_PREFERENCES (↑EcritureDisque")
```

Comme vous l'avez remarqué, aucun chemin d'accès n'est indiqué dans la méthode. Dans ce cas, 4D stocke le fichier directement dans le dossier de la structure (votre base .4DB).

Conclusion



Nous voici au terme de notre découverte et bien qu'il reste encore de nombreux thèmes à découvrir, je pense que vous avez suffisamment appris pour commencer vos développements dans de bonnes conditions.

Mon objectif était de vous faire prendre conscience du potentiel énorme de 4D et de vous donner les moyens d'en tirer parti rapidement.

En complément de ce guide, qui n'a pas la prétention d'être exhaustif, vous trouverez tant sur les forums qu'au cours des formations de solides compléments et réponses à vos questions.

Je vous remercie d'avoir suivi ces différentes leçons avec attention.

Voici pour vous mettre l'eau à la bouche une petite liste des possibilités que nous n'avons pas abordées :

- Gestion des images
- Compilation et génération d'applications
- Publication Web
- Web Services
- XML
- Enumérations
- Styles
- Bulles d'aide
- Bulles d'aide sous forme de variables

Pour aller plus loin

Voici quelques liens et sources d'informations utiles pour compléter votre apprentissage de 4D :

- Le service Formation de 4D SAS : <http://formation.4d.fr>
- Les articles et notes techniques régulièrement mis à jour : <http://www.4d.fr/support/techspace.html>
- Le site officiel de 4D SAS : <http://www.4d.fr>
- L'accès à l'ensemble des manuels de 4D : <http://www.4d.fr/support/docs.html>
- Le forum officiel d'entraide de 4D : <http://forums.4d.fr>
- Un centre de ressources techniques pour 4D : <http://4d.developpez.com>