Master SIS
Professionnel

Option Génie Informatique

Atelier de Génie Logiciel

Illustration avec Windev

[UE GL 7)

Spécialité Génie Logiciel

Erwan TRANVOUEZ
erwan.tranvouez@polytech.univ-mrs.fr

Université
Paul CEZANNE

Ressources (Atelier de) Génie Logiciel

- *Software Engineering : Principles and Practice.* **H. V. VLIET. 2nd ed. J. Wiley & sons Ltd.**
- *Analyse de système orientée-objet et génie logiciel: Concepts, méthodes et application.* **G. LEVESQUE. Chenelière/Mc Graw-Hill.**
- *Génie Logiciel, Jacques PRINTZ Que Sais-Je N° 2956. PUF.*
- *Ingénierie des Systèmes d'Information : MERISE.* **D. NANCI, B. ESPINASSE et al. 4ème ed. Vuibert.**
- *Guide to CASE adoption* **K.S. OAKES, D. SMITH, E. MORRIS, Tech. Report Software Engineering Institute - Carnegie Mellon Univ.**
- *Xtreme Programming :*
  - **http://www.extremeprogramming.org/**
  - **http://c2.com/cgi/wiki?ExtremeProgramming**

Ressources Windev & Outils

- *Liste d'AGL par catégorie :*
  `http://www.cs.queensu.ca/Software-Engineering/toolcat.html`
- *Documentation commerciale des produits mentionnés.*
  - **`http://www.pcsoft.com/windev`**
  - **`http://www.sybase.com/products/internetappdevtools/powerbuilder`**
  - **`http://www.rational.com/products/dstudio`**
  - **`http://www.oraclecom/ip/develop/ids/editions.html`**
  - **`http://www.objecteering.com`**

- *Guide d'AutoFormation*. PC Soft Edition.
- *Site de l 'Association des développeurs Windev*.
  `http://www.windevasso.org`

Plan du support de cours

III - Le W-Language

1. Présentation générale

2. Les composants graphiques disponibles

3. Gestion des événements

Présentation du W-langage

Le W-langage est un langage :

- Procédural
- Orienté Objet
- de 4ème génération (5 depuis Windev 7.5)
  - **qui propose une syntaxe simplifiée** (proche du langage naturel)
  - **qui éloigne certains détails techniques** (gestion mémoire)

Exemple de code en W-Langage

```
A est un entier  //Commentaire
S est une Chaîne = "Bonjour Monde"

Si OuiNon("Voulez vous quitter ?")
       Ferme()
fin
```

Caractéristiques Générales

- **Pas de distinction majuscule/minuscule.**
- **Le code se situe soit au niveau du projet soit au niveau de ses composantes. Il est manipulé via l'Editeur de Source (ES) (F2 ouvre l'éditeur sur le composant graphique sélectionné).**
- **Fonctions prédéfinies du langage classées par type et distinguées par les première lettres de leur nom.**

Ex :

- **Hxxx() :** fonctions de gestion de fichiers (BD).
- **fxxx() :** fonctions de gestions de fichiers (texte, …).
- **ixxx() :** fonctions d'impression.
- **Menuxxx() :** fonctions de gestion de menu…

3. W-Language

Cf. Convention de nommage

Quelques mots clés réservés

- Externe :
  - importation de code en W-Langage contenu dans un fichier texte.
  - Déclaration d'un objet externe (analogie au C)
- MaXXX/MonXXX renvoie des références sur des objets windev afin de les manipuler (ex. accéder à une propriété : MaXXX..Prop)
  - **MaFenetre :** renvoie une « référence » sur la fenêtre courante(*).
  - **MaSource :** idem pour une source de donnée (fichier, vue, requête)
  - **MoiMême :** idem pour composant graphique
  - **MonEtat** : idem pour un Etat (affichage des données contenues dans une source de donnée)
  - …
- Type de donnée, opérateurs …

(*) càd le dernier élément activé/selectionné (ex: composant graphique activé, fenêtre ayant le focus…)

Déclaration de variables (1/4) :
Types de variables disponibles

Nom Type	Valeurs (min ... max)
booléen	Vrai (<> 0) ou faux (0)
entier court	0 … 255
entier	-32 767 … +32767
entier sans signe	0 .. 65 535
entier long	- 2 147 483 647 ... +2 147 483 647
réel	3.4×10^{-38} … $3.4\times10^{+38}$
monétaire	- 604 462 909 807 314 587, 353 087…+ 604 462 909 807 314 587, 353 087
réel double	1.7×10^{-308} … $1.7\times10^{+308}$
Caractère	1 caractère
Chaîne	Chaîne de taille dynamique se redimensionnant toute seule. Max 65536 car.
Chaîne fixe sur n	Chaîne de taille fixe max.32 768 caractères
Chaîne ASCIIZ sur	Chaîne terminée par \0 (pareil que C). Max 65536 car. (\0 compris).

Déclaration de variables (2/4):
syntaxe et exemples

Syntaxe :

```
NomVar est [un/e] Type [=val]
a, b sont [des] Type[s]
Tab est un tableau de <dim1> [(par|,) <dim2>] type
TabDyn est un tableau dynamique de <dim1> type
TabDyn = allouer un tableau dynamique de <Dim1> … Type
//ATTENTION : l'indice d'un Tableau commence à 1
// Impossible d'affecter un tableau par un autre
```

Exemples:

```
Test est un booléen = vrai
Nom est une chaîne
taille est entier
PI est un réel = 3.14
tab est un tableau de 8 par 8 entier
tabN est un tableau dynamique de 10 chaînes
Clients = allouer
taille = dimension(Clients) //renvoie la taille du tableau
Nom = tabN[3]; ou
Dimension(tab, 20) redimensionne le tableau tab (non dynamiques uniquement)
```

Déclaration de variables (3/4):
types complexes

Déclaration de structures :

```
NomStruct est une Structure
   <membre1> est un <Type1>
   …
Fin
```

Utilisation de structures :

```
Var est un NomStruct
```

Pour accéder au champ de la structure utiliser ':'

```
Var:membre1 = 10
```

Exemples:

```
Complexe est une structure
  réelle est un réel
  imaginaire est un réel
Fin
```

```
z est un Complexe
z:réelle=2.5
z:imaginaire=-3
```

Il n'est pas possible d'affecter une structure à une autre.

```
Ex :   z1 est un Complexe
       z1=z2 // génère une erreur
```

Déclaration de variables (4/4):
divers

Déclaration de constantes :

```
CONSTANT
   <const> = <val>
   …
Fin
```

Exemple :

```
CONSTANT
  PI=3.14
  NbClientsMax = 10000

FIN
```

Portée des variables :

Globale : dépend d'où on déclare

- ***Initialisation projet :* visible partout**
- ***Traitement ouverture fenêtre :* dans le code de la fenêtre seulement**

La déclaration se fait à l'aide du mot clé Global

Local : mot clé local optionnel. Peut servir à marquer la fin d'un bloc de déclaration de variables globales.

Portée : limitée au bloc du traitement.

Exemple :

```
Global
  NbClients est un entier
  monNom est une Chaîne
  CONSTANT PI=3.14
Fin
```

Opérateurs (1/2)

Logiques : **`et ou pas`**

Arithmétiques : **`+ - * / ++ -- += -=`**

Comparaisons : **`= <> <= >= > <`**

Chaînes de caractères :

+ concaténation de deux chaînes

++ concaténation d'une chaîne avec elle-même

[[ ]] extraction de chaîne. (commence à 1)

Ex : ch[[i]] , ch[[3 à 10]], ch[[3 à]] , ch[[à 10]] et ch[[3 sur 4]].

~= : égalité souple sur les chaînes de caractères : ne tient pas compte des minuscule/majuscule, espaces avant/après, accents sur minuscules)

~~ : égalité **très** souple. En plus de ~= ne tient pas compte des espaces et caractères de ponctuation dans les chaînes.

Adresse: **&** (renvoie adresse d'une variable hors type dynamique)

Opérateurs (2/2)

Indirection de champs et de rubrique : `{}`

Convertit une chaîne en un nom de rubrique (variable désignant un objet graphique par exemple).

MoiMême : renvoie le champ en
MaFenêtre, MonEtat parlent d'eux même !

Exemples :

```
{"LIBELLE"}= "Mon libellé"
```

{nomS+"..Libelle"} = "Saisir : " // où nomS peut contenir le nom d'un champ de saisie

Opérateurs et ponctuation :

`[ ]`	**accède à un indice d'un tableau ou d'une liste**
`, . : ; ]`	**séparateurs de variables**
`. .`	**accède à la propriété d'un champs**
`. . .`	**équivalent du \ en C. S'utilise après , (liste paramètres) ou + (utile pour la concaténation notamment de chaînes longues)**
`//`	**commentaire sur une ligne**

Structure de contrôle (1/3) : Conditions

Instruction conditionnelle SI : **équivalent du** if-else **en C**

```
Si <condition> alors
      <action si vrai>
[Sinon
      <action si faux> ]
Fin
Si <condition> alors <action>
```

```
Si i<10 et modifier alors i++
```

Alors doit être sur la même ligne que Si

Instruction conditionnelle SELON : **généralisation du** switch-case **en C**

```
Selon <expres1> // peut valoir une chaine
      Cas <exp2> : <action1> // 1 seule ligne
      Cas <exp3> [,<exp3a> …]
            <action2>
      Autres Cas // ou Autre cas
            <action par défaut>
Fin
```

Dans le cas de chaine de caractère La comparaison tient compte de la casse des caractères.

Structure de contrôle (2/3) : Boucles

Instruction TantQue : **équivalent du** while **en C**

```
TantQue <condition>
     <actions>
 Fin
```

```
TantQue i>10
        info("i="+i)
        i++
Fin
```

"Sortir" peut forcer la sortie de la boucle

Instruction Boucle : **équivalent à** 'TantQue vrai'

```
Boucle <condition>          //alors forcer la sortie de la boucle
     <actions>              // avec sortir
Fin
```

Retour : permet de quitter le traitement en cours (procédure ou gestion événement)

Structure de contrôle (3/3) : Boucles

Instruction Pour : **équivalent du** for **en C**

```
Pour <condition> = <Vini> à <VFin> [pas <x>] // par defaut 1
     <actions>                          sortie de la boucle
Fin
```

```
pour i=0 à 10
     info(i)
fin
```

```
pour i=0 à 10 pas 2
        info(i)
fin
```

Instruction Goto :

se branche sur une étiquette donnée (ne pas utiliser mot réservés! Ex Fin)

```
Goto <Etiq1>
  // ...
     <Etiq1> : //Code
```

```
goto fini:
//...
fini: info("fini")
```

Les fonctions et procédures avec le W-Langage

- Comme en Pascal, une fonction se distingue d'une procédure en ce qu'elle renvoie une valeur avec `Renvoyer`.
- Par défaut les variables sont passées par adresse! Pour passer les variables par valeur il faut entourer la/es variable/s entre parenthèses. (Ex: **`fctn( (a) )`**).
- La portée globale (projet) et locale (fenêtre) est également applicable aux procédures.
- La définition/création d'une procédure globale (Ctrl+F8 ou Shift F4 depuis l'ES(1)) /procédure locale (F8 ou F4 depuis l'ES(1)) à une fenêtre peut être simplement gérée via le menu Code.
- Le code de ces procédures est également accessible via le Kouglov dans l'onglet Procédure du Projet (procédure globale) ou d'une fenêtre (procédure locale).

3. W-Language

(1) Editeur de Source

Déclaration d'une procédure

```
Procedure <nom> ( [param1 [, param2] ] )
      <actions>
      [Renvoyer <valeur>]
```

- Les paramètres ne sont pas typés. Il est cependant possible de connaître le type des variables dynamiquement avec `typeVariable` (renvoie un entier).
- Définir une procédure dans le code de traitement d'ouverture d'une fenêtre (si possible nommée comme la fenêtre) permet de paramétrer l'ouverture de la fenêtre.
- Il n'est pas possible de passer un tableau en paramètre par valeur.
- Paramètres optionnels :

```
Procedure <nom> ( param1, param2, param3=val_defaut, param4=5)
      ...
```

Ils sont placés en dernier et doivent avoir une valeur par défaut. Pour forcer une valeur à `param4` il faudra en donner une à `param3`.

Appel d'une procédure

```
Procedure bonjour ( nom, Titre="?" )
        ch est une chaine =nom
        info("titre="+titre,"nom="+nom)
        selon titre
                cas "?", "Mle", "Mme", "M."
                        si titre<>"?" alors
                                ch=titre+ch
                                info("Bonjour "+ch)
                                renvoyer vrai
                cas autre : renvoyer faux
        fin
```

Nom aurait pu être déclaré optionnel

Rappel : les variables sont passées par adresse !!! D'où la nécessité de copier nom dans ch.

Appel de procédure : **Nom_procedure([Parametres])**

```
bonjour("toto")       affiche Bonjour ? Toto
bonjour("toto","M.") affiche Bonjour M. Toto
bonjour("toto","a")  retourne Faux
```

3. W-Language

Quelques principes de programmation

Définir des règles de codage :

- ***Structures des programmes, classes, fonctions …***
  - *Ex : limiter la portée des fonctions. Une fonction n'étant pas appelée en dehors d'une fenêtre doit être déclarée Locale à la fenêtre.*
- ***Regles de nommage***
  - *La facilité d'écriture de code peut conduire à du code mal écrit …*
  - *… et donc un programme source d'erreur*
  - *La règle peut être adaptée pourvu qu'elle soit suivie tout le long du projet*
  - *Ex : Régles de nommage proposée par :*

    `http://rbesset.net/modules/icontent/index.php?page=12`

3. W-Language

Cf. détails Ci après

Exemple de regle de nommage

- **Nom descriptif:**
  - nommer du plus général au plus particulier.
  - Ex: MonnaieNomComplet, MonnaieNomAbr …

- **Préfixes de portées :**
  - Globale projet : **g**
  - Globale fenêtre : g**f**
  - Paramètres : **p**
  - Locale : **''** (rien)

- **Préfixes de type :**
  - Chaîne caractères (et mémos textes) : **c**
  - Numérique / Monétaire : **n**
  - Memo binaire : **b**
  - Booléen : **b**
  - Structure : **s**
  - Objet : **o**
  - Variant : **v**
  - Zones mémoires : **m**
  - ...

- **Noms variables : PréfixePortée + PréfixeType + NomVariable**
- **Noms fonctions : PréfixePortée + NomFonction**

IV - Programmation graphique avec Windev

1. Présentation générale
2. Les composants graphiques
3. Gestion des évenéments

Propriétés communes à tous les composants

Propriété	Description
..Type	**Type du composant (ex de valeur : typBouton, typFen, typSTATIC, …)**
..Hauteur	**Largeur du composant graphique**
..Largeur	**Longueur du composant graphique**
..Couleur	**Couleur du texte**
..Bulle	**Texte de la bulle d'aide obtenue lorsque la souris reste sur un composant.**
..Valeur	**Varie selon le composant (ex pour saisie correspond au texte saisi par l'utilisateur). Cette valeur est directement accessible avec le nom du composant.**
..Etat	**Pour les champs peut valoir : *actif* (réagit aux événements), *Grisé* (visible mais désactivé), *AffichageSeulement* (pas modifiable), *AffichageSansSélection*.**
..Visible	**Indique si la fenêtre est visible. Vaut vrai ou faux. Peut être modifié => modifie l'affichage du composant. Restriction sur certains composants**

4. Programmation graphique — **Tous les composants disposent d'un traitement d'initialisation.**

Liste et propriétés spécifiques des composants graphiques standard (1/2)

Champs	Propriété	Description
Fenêtre Fenêtre Windows. ***Evénements gérés**:* Ouverture et Fermeture, Prise et perte de focus Modification de la taille	CurseurSouris	**Change l'apparence du curseur**
	Libellé	**Nom de la fenêtre**
Libellé Texte non modifiable libre *Evénements gérés :* **aucun**	Valeur	**Texte affiché dans le libellé**
Saisie **Zone de saisie** *Evénements gérés :* Entrée et Sortie du champ, Modification contenu.	Valeur	**Texte entré dans la zone de saisie**
	MasqueSaisie	**Masque de saisie appliqué. Ex maskINSEE**
Bouton : **bouton** *Evénements gérés :* clic souris	Libellé	**Libellé du bouton.**

4. Programmation graphique

Les évènements indiqués sont ceux gérés par défauts. D'autres sont cependant possible (clic bouton milieu par exemple)

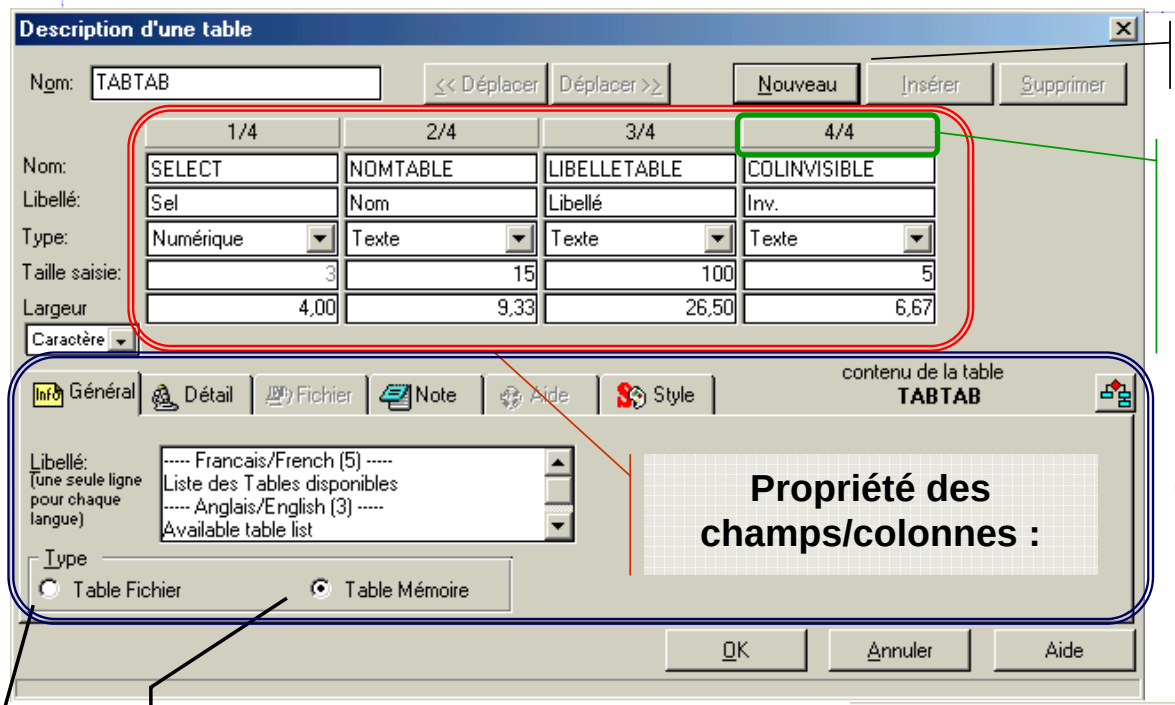
Liste et propriétés spécifiques des composants graphiques standard (2/2)

Champs	Propriétés	Description
Liste :  Liste déroulante. ***Evénements gérés :*** Entrée et Sortie, Modification.	Valeur	**Position dans la liste de l'item sélectionné (nb entier)**
	Occurence	**Nombre de lignes (ie éléments)**
	nomListe[i]	**Renvoie le texte de l'item I de la liste nomListe**
	nomListe = I	**Sélectionne l'item I de la liste nomListe**
Combo Box :  Liste déroulante sur 1 ligne	Idem que Liste	
Interrupteur :  Case à cocher/Liste d'options ***Evénements gérés :*** idem	Libellé	**Libéllé qualifiant la zone d'options**
	opt[i] opt[i]..Libellé	**Vaut 0/faux ou 1/vrai selon que l'option opt[i] est cochée ou non** **Libéllé de l'option i**
	Occurence	**Nombre de cases à cocher/options présentes**
Sélecteur :  Liste de choix unique ***Evénements gérés :*** idem	Valeur	**Vaut un numéro correspondant au choix sélectionné**
	Libellé	**Libellé de la liste d'options**
	select[i]..Libellé select[select]	**Libellé du i^{ème} choix** **Libéllé du choix en cours**

4. Programmation graphique

Tous les composants disposent d'un traitement d'initialisation.

Présentation des tables Windev 5.5



Créer un nouveau champ

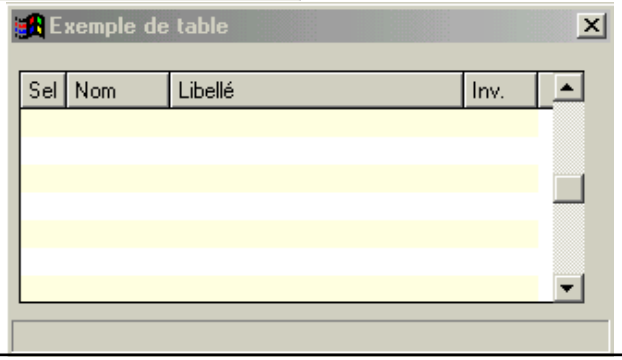
En cliquant sur cette zone les onglets de propriétés portent spécifiquement sur une colonne. Par exemple ajout d'un type "calculé" .

Onglets de propriétés habituels : général, détail, note, style et code.

Propriété des champs/colonnes :

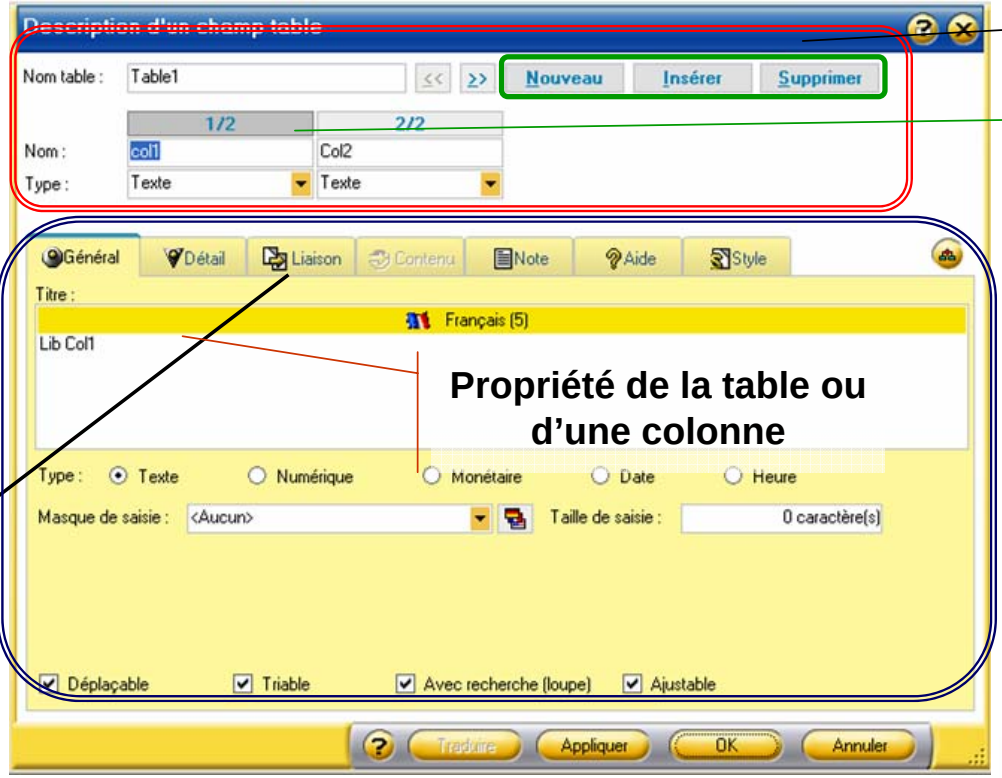
liée à une liste de valeur stockée en mémoire

liée à un fichier de l'analyse (entité du MLD)



4. Programmation graphique

Présentation des tables Windev 7.x



Créer un nouveau champ

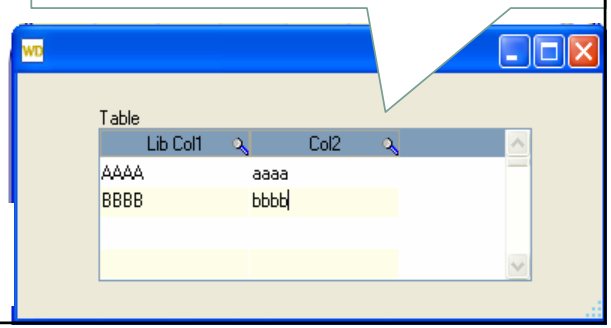
En cliquant sur cette zone les onglets de propriétés portent spécifiquement sur une colonne.

Onglets de propriétés habituels : général, détail, note, style et code.

Propriété de la table ou d'une colonne

Propriétés supplémentaire si la table est liée à un fichier de l'analyse (entité du MLD) ou une requête

Ajout dans la ligne des libellés de colonnes d'une fonction de tri (clic sur libellé) et de recherche de valeur (clic sur loupe)



Menu contextuel propose exporter contenu table sous format Excel ou XML

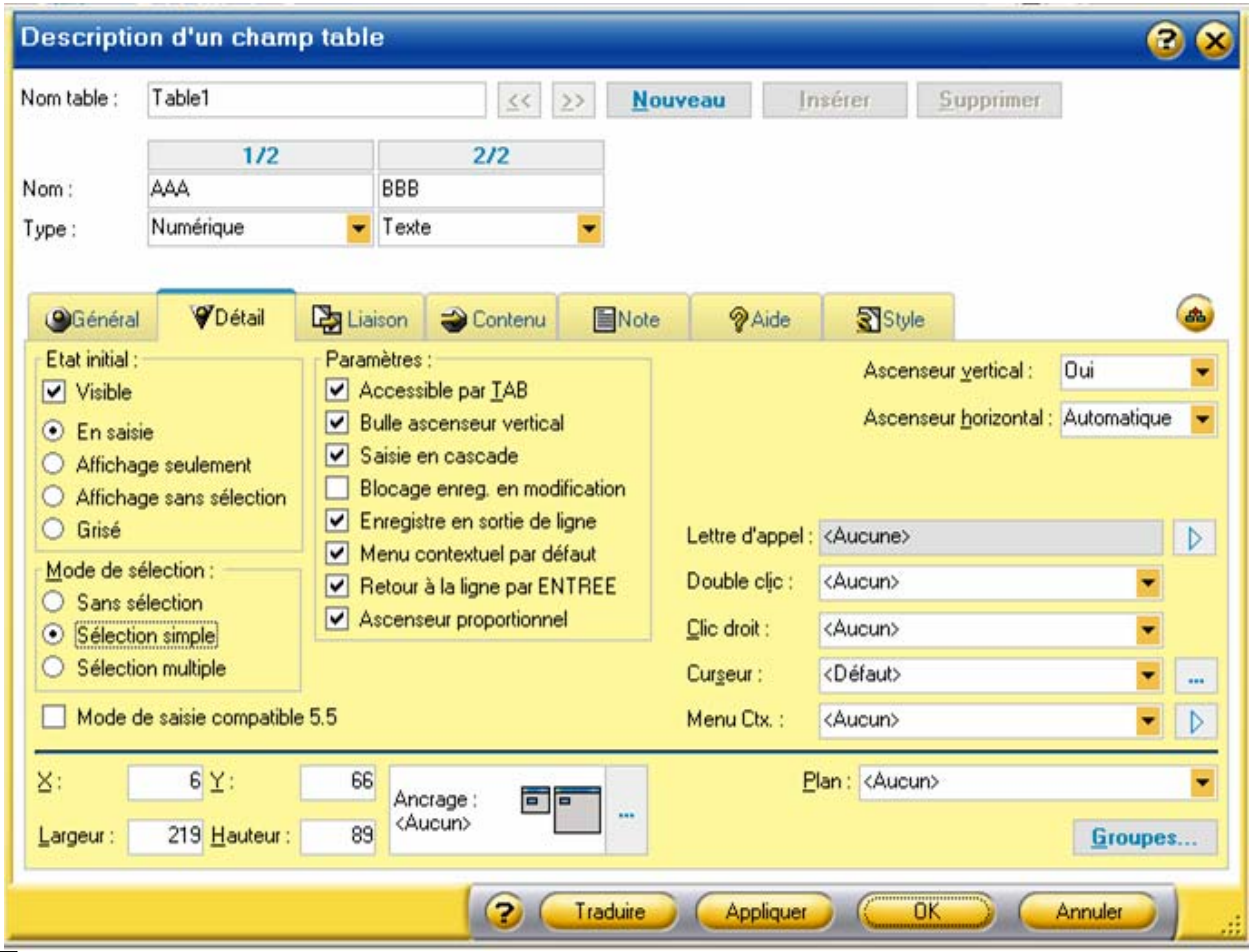
4. Programmation graphique

Propriétés d'une table

Propriété	Description
..Occurrence	Nombre de lignes (valeurs) dans la table
..Vide	Eq. à (nomTable..Occurence=0)
..Nouveau	Indique si la ligne en cours vient d'être crée
..Modifie	Indique si la ligne courante ou le champ a été modifié
..Filtre	Pour les tables Fichier permet de gérer les contraintes simples sur les données liées.
..Triée	Vrai ou faux selon que la liste affiche les données triées ou non. Modifiable.

4. Programmation graphique

Détail d'une table



4. Programmation graphique

Evénements d'une table

Evénement	Description
(Fin d') Initialisation	Appelé lors de l'ouverture de la fenêtre contenant la table. (Fin d') utilisé pour les tables fichiers (sépare le code fichier du code table).
Entrée	Entrée dans la table ie prise de focus(Clic à l'intérieur de la table ou sélection via tabulation ou raccourci clavier …)
Sortie	Sélection d'un autre champ (sortie de focus).
Entrée en saisie d'1 ligne de la table	Le double Clic dans une table sur une ligne… Permet par exemple de sauvegarder avant modification pour pouvoir l'annuler.
Affichage d'1 ligne	Appelé lors de l'ajout d'une ligne ou lors de l'utilisation de l'ascenseur. Peut être également utilisée pour mettre la formule de calcul des champs calculés
Sortie dans Ligne	Sortie de ligne (sic) vers une autre ligne ou traitement appelé lors de la sortie de la table avant l'exécution du traitement sortie de la table.
Sélection d'un Ligne	Sélection (clic) sans saisie d'une ligne.

4. Programmation graphique …auxquels s'ajoutent les traitement liés aux colonnes ((E/S)

Manipulation d'une table

- Soit `ma_table` une table mémoire composée de deux colonnes `Nom` et `Age`.
  - **`ma_table:`** vaut la position de la ligne courante (i.e. active) *Ex* : **`1`**
  - **`ma_table[i]:`** vaut la valeur de la i^{ème} ligne de **`ma_table`** sous la forme d'une chaîne.E*x:*
    `ma_table[ma_table]` peut valoir : **`"Toto 10"`** (valeurs séparées par **TAB**)
  - **`Nom:`** vaut la valeur du champ **`nom`** de la ligne en cours.
    Ex : **`Nom`** vaut **`"toto"`**
  - **`Nom[i]:`** vaut la valeur du champ `nom` de la ligne `i`.
    Ex : **`Age[2]`** vaut **`3`**

Procédure de manipulation d'une table 1/2

- Quelques procédures manipulant les tables
  - **`TablePosition(<table>[,<Indice>])` :** positionne à la ligne **`<Indice>`** dans la table **`<table>`** (sans le selectionner). Si **`<Indice>`** n'est pas renseigné, retourne la ligne du 1er élément **affiché**.
  - **`TableSelectPlus(<table>,<Indice>)` :** sélectionne le **`<Indice>`**ème élément de **`<table>`**
  - **`TableAjoute(<table>,[<valeur>])` :** Ajoute 1 ligne en fin de table ou si la liste est triée à sa position par rapport au critère de tri).

    `<Valeur>` est de la forme **`<valChamp1>+TAB+<valChamp2>+…`**

    Ex :
    ```
    TableAjoute(Table)
    TableAjoute(Table,"Toto"+TAB+10)
    ```

    TAB sert de délimiteur de champs

    ajoute une ligne vide

    ajoute une ligne avec la valeur Toto dans le 1er champ et 10 dans le 2ième

    Pour les tables mémoire, retourne un booléen indiquant le succès de l'ajout.
  - **`TableAjouteLigne(<table>,<val1>, …, <valn>)` :** Ajoute 1 ligne en fin de table

    `<Vali>` est la valeur (compatible) que doît prendre le ième champ.

    La valeur par défaut est la chaîne vide ou 0.

Procédure de manipulation d'une table 1/2

`TableInsere(<table>,[<valeur>],<indice>)` : Ajoute 1 ligne à **`<indice>`**

- **`<indice>`** : entier indiquant la position désirée. Si indice n'est pas renseigné, place à la position courante ou à la fin de la table si aucune ligne est sélectionnée.
- **Idem que `TableAjoute` pour la valeur de retour.**

- **`TableAffiche(<table>)` :** Réaffiche le contenu de la table (recalcul)
- **`TableCherche(<champ>,<valeur>[,<Type>[,<Début>]])` :** Recherche **`<valeur>`** dans la colonne **`<champ>`** à partir de **`<Début>`** en effectuant une comparaison stricte **`(<Type>=Vrai).`** Si **`<Type>`** vaut **`vrai`** revient à chercher toutes les valeurs **`>=`** à **`<Valeur>.`**

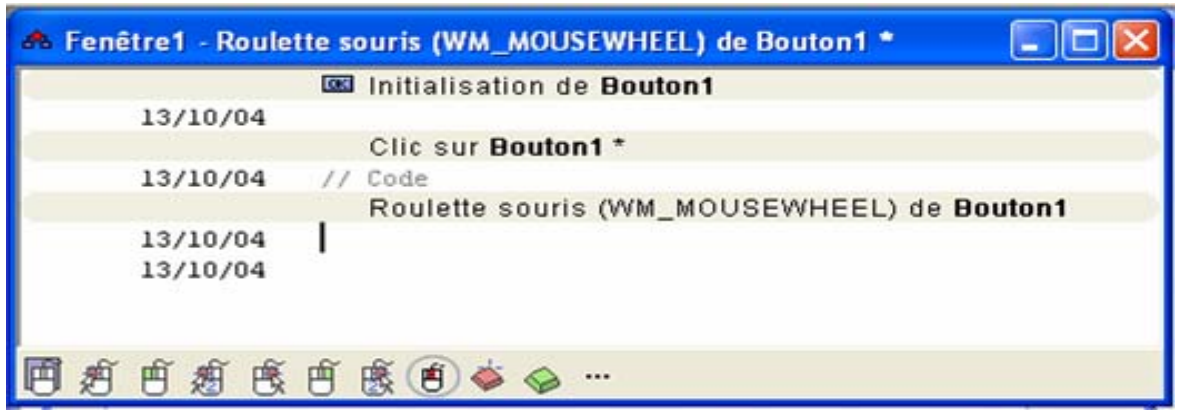
  `TableCherche` retourne le numéro de ligne si **`<valeur>`** est trouvé ou **`-1`** sinon.
- **`TableTrie(<table>,<TriCol1>[,<TriColN>])` :** Tri le contenu de la table

avec : **`<TriCol1>`** = <Sens><NomCol> ou Sens vaut – pour croissant et + pour décroissant et <Nomcol> correspond au nom de la colonne a trier.

renvoie vrai ou faux selon que le tri ait été effectué ou non.

Evénement gérés par défaut

- **L'évenement le plus fréquemment utilisé est naturellement le clic souris.**
- **Depuis la version 7 ajout de nouveaux évenements : autres boutons (droit, milieu, roulette), touches ainsi que d'autres pouvant être ajouté. Ce qui avant nécessitait d'intercepter directement les évenements Windows.**



Exécuter directement le code associé à un événement

- Utilisation de la méthode Execute :

```
Execute("<nom>.. <type_evt>")
```

–<nom> désigne un champ ou une fenêtre
–<type_evt> désigne l'événement

- Exemple d'événements associés à un champ ou à une fenêtre:

Type_evt	Evénement Champ
Clic	**Clic d'un bouton**
Sortie	**Sortie d'un champs**
Entree	**Entrée dans un champs**
Modifie	**À chaque modification**
Initialisation	**Initialisation du champs (revient à réinitialiser)**

Type_evt	Evénement fenêtre
NF	**Ouverture de la fenêtre**
FEF	**Fermeture de la fenêtre**
PRF	**Prise de focus**
PEF	**Perte de focus**
MOD	**Modification de la taille**

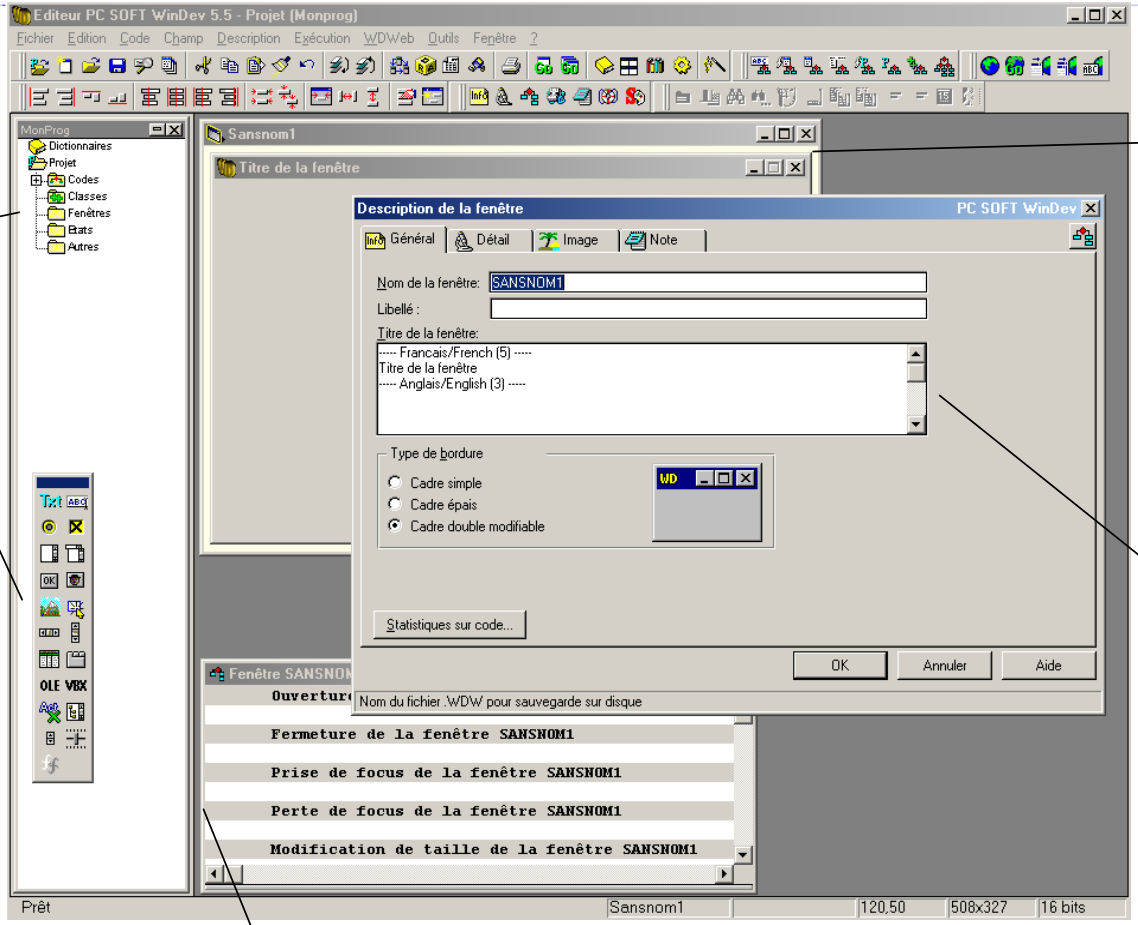
- **Exemples d'utilisation :**

```
execute("Ok_btn..clic")
execute("saisie..modification")
execute("fen..mod")
execute("saisie..pef")
```

Annexes

Ecrans Windev 5.x

L'éditeur de fenêtre Windev 5.5



TreeView

Liste des objets composant le projet.

Champs

Objets graphiques pouvant être inséré dans la fenêtre (Bouton, zone de texte, …)

Fenêtre

Composant graphique en cours de conception

Propriété

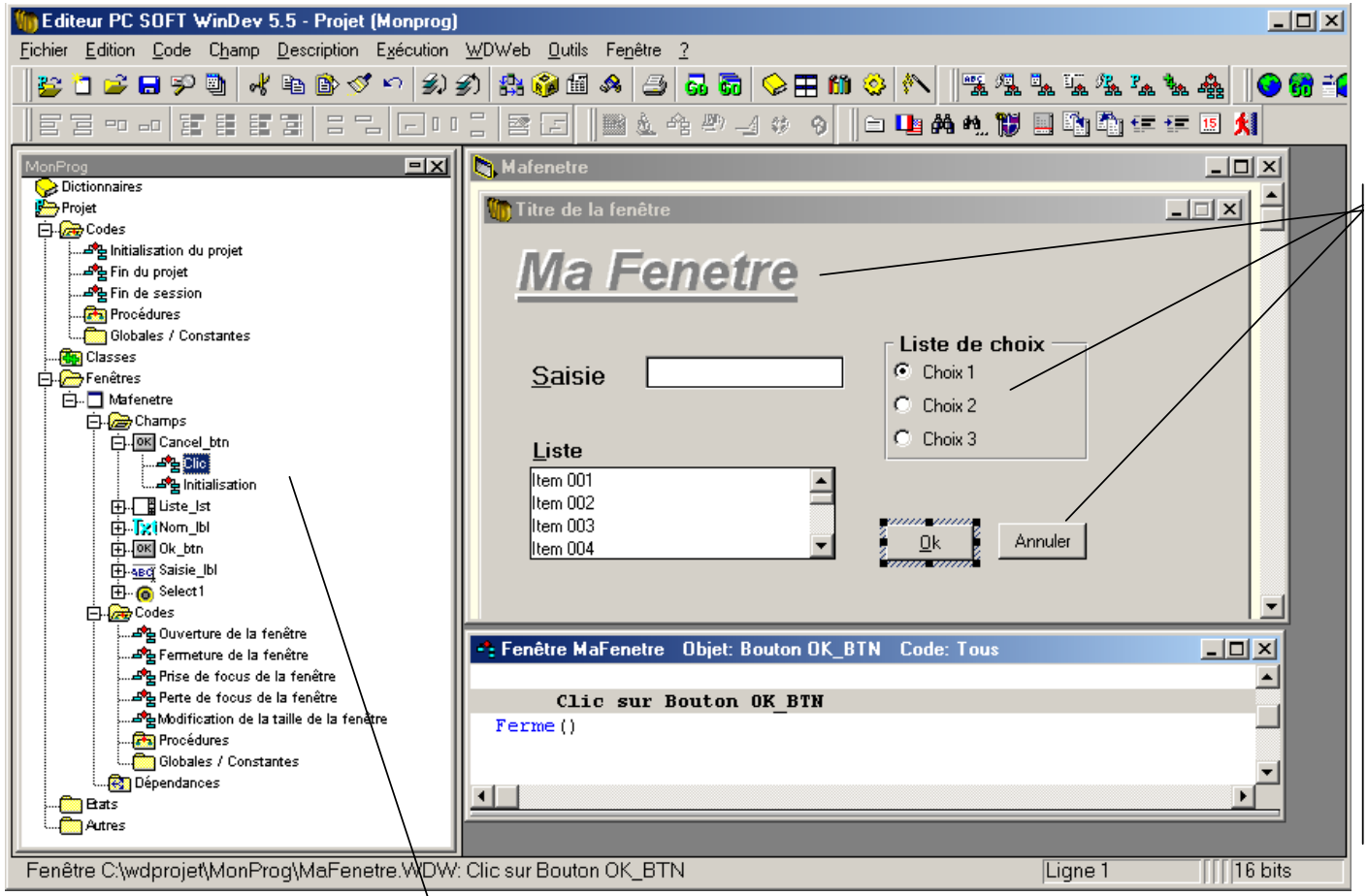
Edition des propriété de la fenêtre.

Code

Listes des événements gérés par le composant graphique et leur comportement associé défini avec le W Langage.

2. 1er pas avec Windev

Exemple de conception de fenêtre (Windev 5.5)



Chaque composant graphique est identifié par un nom dont la portée est celle de la fenêtre dans laquelle il s'inscrit.

Les propriétés du composant seront accédées à partir de ce nom. Ex :

`Label..Valeur:` contient la valeur saisie dans le champs de saisie.
`Label..Libellé:` contient le texte du libellé.

Chaque composant listé dans le **treeview** est décrit par son type par une icône (ex : bouton) et les événements gérés (et le code associé).

2. 1er pas avec Windev

Le champ de saisie Windev 5.5

Permet à l'utilisateur de transmettre des informations à l'application