

Editions ENI

# **WinDev 12**

**Les fondamentaux du développement  
avec Windev ...**

Collection  
Ressources Informatiques

*Extrait*

WinDev propose un langage qui comporte classiquement des variables de plusieurs types, des opérateurs ou des instructions qui offrent la possibilité d'écrire des procédures ou des fonctions. Il dispose de plusieurs familles de fonctions standard pour faciliter la tâche du développeur.

Les instructions et les fonctions peuvent être écrites en français ou en anglais. Chaque nom de fonction a son équivalent en anglais.

Nous allons décrire synthétiquement ces éléments de manière à être capable de les utiliser pour les besoins courants.

Rappelons auparavant quelques principes sur l'utilisation de variables.

### A. Variable, portée ou domaine de visibilité

Pour avoir un code le plus fiable possible, il convient de déclarer les variables qui vont être utilisées dans un traitement. Cela permet au compilateur de signaler l'inexistence de déclaration ou de vérifier que certaines opérations d'affectation ou d'emploi dans des fonctions, sont possibles.

Une variable a donc un type. Il est de bonne pratique d'initialiser la variable à sa déclaration. WinDev nous facilite la tâche en initialisant une variable à une valeur naturelle pour le type : zéro pour les numériques, une chaîne vide pour le type caractère ou chaîne.

Une fois déclarée, une variable a "un espace de vie" qui est plus ou moins étendu selon qu'en plus du type nous précisons que la variable est locale ou globale.

Une variable est par défaut locale ; c'est-à-dire qu'elle n'existe que dans le traitement dans lequel elle apparaît : section de traitement, procédure, fonction. Ainsi, si le traitement appelle une fonction, la variable ne sera pas connue du code de la fonction appelée. Le seul et le meilleur moyen est de passer cette variable en paramètre de la fonction.

Si la variable est déclarée comme globale, elle est accessible par les traitements déclenchés depuis le traitement où elle a été déclarée.

Pour déclarer des variables comme étant globales, le mot clef **GLOBAL** doit être indiqué avant la liste des déclarations des variables concernées. De même, pour les déclarer locales, le mot clef **LOCAL** est à placer avant la liste correspondante.

```
Global
CodeClient, NumFacture sont des entiers
Local
cNom, cPrenom sont des chaînes
```

Si une variable est déclarée globale dans le code d'initialisation d'un projet, elle est connue de tous les traitements du projet.

Si une variable est déclarée globale dans une fenêtre, elle est connue de tous les traitements découlant de cette fenêtre.

➤ Les champs d'une fenêtre réagissent comme des variables globales pour la fenêtre.

Si une variable est appelée du même nom qu'une variable définie dans un traitement de plus haut niveau, la variable du traitement courant "masque" la variable de plus haut niveau. C'est le contenu de la variable du traitement en cours qui est utilisé.

## B. Les constantes

WinDev fournit différents fichiers qui comportent des listes de constantes. Les constantes Windows sont déclarées dans le fichier WINCONST.WL. Les limites des différents types de variables de WinDev sont dans le fichier LIMITES.WSF.

Nous pouvons créer nos propres constantes par une déclaration de la forme :

```
Constante
  CtEnCours=1
  CtResilie=2
Fin
```

Comme pour les variables, les constantes ne sont connues que des traitements déclenchés en dessous du traitement où elles ont été déclarées.

Si une variable est déclarée avec le même nom qu'une constante, c'est la variable qui est utilisée dans tout son domaine de visibilité. Elle masque la constante de même nom dans son espace.

L'intérêt d'utiliser une constante dans le code, au lieu d'une valeur, se manifeste s'il faut modifier la valeur initiale.

Avec l'emploi d'une constante il suffit de modifier la valeur dans l'instruction de déclaration, sinon cette valeur est à rectifier dans toutes les lignes de code où elle a été utilisée.

## C. Les types de variable

WinDev propose des types simples de variables et des types avancés.

Les types simples sont : booléen, entier, monétaire, réel, chaîne, buffer, date, heure, dateHeure, durée, variant, numérique.

Les types avancés comprennent les tableaux, les tableaux dynamiques, les tableaux fixes, les variables composées, les buffers dimensionnés, les structures et des structures particulières qui servent à décrire les fichiers, les relations et les rubriques, de manière temporaire.

Pour déclarer une ou des variables d'un même type, la syntaxe est la suivante :

```
<Variable1>, <Variable 2> sont des <Type de variable>
```

On peut initialiser une variable lors de la déclaration, sous la forme :

```
<Variable1> est un(e) <Type de variable> =<Valeur>
```

On ne peut pas initialiser plusieurs variables dans la ligne de déclaration. Seule, la dernière variable citée est initialisée si nous mettons une valeur au bout d'une ligne de déclaration avec plusieurs variables.

Les types ont des valeurs maximales et minimales ! Si une variable dépasse ces limites, un message d'erreur est généré en mode test mais en exécution, la valeur est fautive et il n'y a pas de message.

Les valeurs limites sont récapitulées dans le fichier personnel\externe\limites.wl.

### 1. Booléen

Une variable de type booléen ne prend que deux valeurs : vrai ou faux. La valeur Faux vaut zéro. La valeur Vrai est différente de zéro.

**B**Valeur est un booléen.

### 2. Entier

Les entiers de WinDev sont codés par défaut sur 4 octets, ce qui permet de manipuler des entiers assez grands avec le type de base.

Sinon nous pouvons choisir les types suivants :

Entier	-2*10 <sup>9</sup> à 2*10 <sup>9</sup>
Entier sur 1 octet	-128 à 127 (incluses)
Entier sur 2 octets	-32 768 à 32 767 (incluses)
Entier sur 8 octets	-9*10 <sup>18</sup> à +9*10 <sup>18</sup>
Entier non signé	0 à 4*10 <sup>9</sup> ou 2*10 <sup>10</sup>
Entier sur 1 octet non signé	0 à 255
Entier sur 2 octets non signé	0 à 65 535
Entier sur 4 octets non signé	0 à 4*10 <sup>9</sup>
Entier sur 8 octets non signé	0 à +9*10 <sup>19</sup>

Le type **Entier Système** est disponible et doit être utilisé dans le cas de l'utilisation des fonctions API ou AppelDll32, pour permettre à WinDev d'adapter les exécutables à un environnement en 32 bits ou en 64 bits.

### 3. Monétaire

Le type monétaire est un type important dans WinDev car il garantit la précision des calculs jusqu'à 6 décimales. Il est codé sur 10 octets et les valeurs extrêmes vont de + à - 600 mille milliards environ.

**M**Montant est un monétaire.

## 4. Réel

WinDev propose des réels sur 4 octets avec 6 chiffres significatifs ou 8 octets avec 15 chiffres significatifs. On pourra préférer utiliser ce type si la précision des décimales n'est pas importante et pour économiser un peu de place.

## 5. Chaîne

Les types usuels sont le caractère, pour des chaînes d'un caractère ou le type chaîne qui a une longueur dynamique pouvant aller jusqu'à plusieurs millions d'octets.

D'autres types existent pour la comptabilité avec différents langages : Turbo Pascal, Visual Basic et les API Windows.

Par défaut, une variable de type chaîne est de longueur nulle. Elle est vide.

## 6. Buffer

Ce type de variable définit une zone mémoire qui peut accueillir des valeurs binaires. C'est-à-dire que la variable peut avoir toutes les valeurs de zéro à 255. Dans ce type, la valeur zéro ne représente pas le chiffre zéro mais bien la première des 256 possibilités de codage qu'offre un octet. Le chiffre zéro est représenté par la valeur 48 (en notation décimale). Sa taille n'est pas fixée.

Il existe un type avancé, **Buffer sur**, qui permet de dimensionner ce type de variables.

## 7. Date, Heure, DateHeure, Durée

Le type **Date** facilite la gestion des dates. La date est stockée sous forme AAAAMMJJ. Les opérations d'ajout ou de retranchement de jour rendent des dates valides automatiquement. Le type Date confère trois propriétés, de type entier, aux variables de ce type : l'**année**, le **mois**, le **jour**.

La plage de dates gérée va de 01/01/0001 à 31/12/9999.

*Exemple des propriétés du type date*

```
dDate est une Date = ''20021210''
Info("annee",ddate..annee) donne 2002
Info("mois",ddate..mois) donne 12
Info("jour",ddate..jour) donne 10
```

➤ Par défaut, une variable date est égale à la date du jour.

Le type **heure** permet de gérer les heures de 00:00 à 23:59. Ce type a également des propriétés : **Heure**, **Minute**, **Seconde**, **Milliseconde**.

### *Exemple des propriétés du type Heure*

```
Par exemple : hMonHeure est une heure = ''20255912''
Info("heure",hMonHeure..heure) donne 20
Info("minute",hMonHeure..minute) donne 25
Info("seconde",hMonHeure..seconde) donne 59
Info("milliemes",hMonHeure..milliseconde) donne 12
```

Le type **DateHeure** a les propriétés des types Date et Heure et les propriétés **PartieDate** qui renvoie ou modifie la date et **PartieHeure** équivalente pour l'heure.

Le type **Durée** permet de gérer facilement des différences entre variables temporelles de type Date ou Heure. Une durée a les propriétés **jour**, **heure**, **minute**, **seconde**, **milliseconde**.

```
dDateDemainMidi est une dateheure = EntierVersDate(DateVersEntier
(Datesys()+1)+ "12000000
dDateMaintenant est une dateheure =datesys()+heuresys()
// supposons avant midi, 10h 59 et 40 secondes pile.
drDiciLa est une duree = dDateDemainMidi-dDateMaintenant
drDicila..jour=1 Info("ecart jour",drDicila..jour) donne 1
drDicila..heure=1 Info("ecart heure", drDicila..heure) donne 1
drDicila..minute=0 Info("ecart minute", drDicila..minute) donne 1
drDicila..seconde=20 Info("ecart seconde", drDicila..seconde) donne 20
drDicila..milliseconde=0 Info("ecart milliseconde", drDicila..millise-
conde) donne 991
```

## 8. Numérique

Ce type permet de garantir une précision sur la partie décimale de valeurs réelles.

Un numérique gère 38 chiffres répartis entre 32 au maximum pour la partie entière et 6 au maximum pour la partie décimale. La précision est assurée sur 6 décimales.

## 9. Variant

Ce type permet d'accueillir n'importe quel autre type de variable. Il permet également des interactions avec les objets **ActiveX** et la programmation **OLE Automation**.

La fonction **TypeVar ()** permet de connaître le type de variable qui a été stocké dans un type Variant.

## 10. Constante Null

Cette constante permet :

- De savoir si une variante de type Variant est initialisée.
- D'affecter la valeur par défaut du type de la variable, 0 pour les numériques, une chaîne vide pour les chaînes.

- De supprimer une référence à un objet dynamique et de libérer l'espace mémoire correspondant si cet objet n'est plus référencé.
- D'initialiser une rubrique Hyper File avec la valeur par défaut indiquée dans la définition de la rubrique dans l'analyse.
- D'initialiser à vide un champ de saisie lorsque la case à cocher **Null si vide** est cochée dans l'onglet **Détail** de définition. Ceci permet de renvoyer un champ vide : il sera égal à Null. Cette valeur est détectée par le gestionnaire de requête qui ne tiendra pas compte de paramètres dont la valeur est Null.

## 11. Types de variable avancés

Les types avancés sont constitués par la composition de variables de type de base qui peuvent être manipulées globalement par un seul nom.

Ce sont les tableaux, les variables composées, les structures, les objets automation, les sources de données, les descriptions de fichier, les descriptions de liaison, les descriptions de rubriques et les polices.

### a. Tableaux

Les tableaux sont évidemment très utiles, au moins pour gérer des groupes de cases à cocher ou des listes.

Un tableau peut être simple, fixe ou dynamique.

Le type simple est déclaré en taille, en indiquant le nombre d'éléments par dimension (jusqu'à dix dimensions). Le nombre d'éléments par dimension peut être modifié par la fonction **Dimension**. Celle-ci ne peut pas modifier le nombre de dimensions.

Ce type peut être utilisé dans une structure. Un tableau ne peut pas contenir de tableau.

Le type dynamique permet de créer un tableau avec un nombre d'éléments par dimension qui peut être calculé avant de dimensionner le tableau réellement. En fait, il convient de le déclarer sans dimension puis de l'allouer en mémoire avec les dimensions paramétrées.

Le tableau fixe a des dimensions qui ne pourront pas être modifiées lors de l'exécution. Ce type de tableau doit être employé pour correspondre avec des API.

Les syntaxes sont les suivantes :

```
TBS est un tableau de 5, 7, 3 monétaires // tableau simple avec 3 dimensions
TBD est un tableau dynamique
TBD = Allouer un tableau dynamique de 5, 7, 2 chaînes de caractères
TBF est un tableau fixe de 3,3 entiers
```

Les tableaux peuvent être passés en paramètre à des fonctions ou procédures. Il suffit de donner le nom du tableau comme paramètre.

```
MaFonction(NomDuTableau).
```

Editions ENI

# **WinDev 12**

**Entraînez-vous à développer  
10 fois plus vite**

Collection  
Les TP Informatiques

*Extrait*

## Chapitre 4 : Programmation avancée

● **Durée** : 5 heures 20

● **Mots-clés** : classes, objets, UML, POO, XML, compilation, cryptage, requêtes, SQL, Sockets

● **Objectif**

Le W-Langage est un langage des plus élaborés. Il est ouvert à toutes les méthodes de programmation avancées, telles que la programmation orientée objet ou la génération dynamique de code source. Les classes livrées permettent de faire ses premiers pas en la matière, offrant la possibilité au développeur de concevoir des applications complexes et évolutives. Une bonne connaissance du langage est indispensable au suivi des exercices proposés ci-après. Les TP de ce chapitre ont pour but de donner aux programmeurs WinDev un aperçu des fonctionnalités évoluées mises à leur disposition.

À l'issue de ce chapitre, vous serez capable de :

- ▶▶ Piloter des classes complexes, faisant appel aux API Windows pour représenter des objets en 3D et les manipuler.
- ▶▶ Construire un "parser" XML, facilitant les échanges de données informatisées entre systèmes hétérogènes.
- ▶▶ Générer à la volée du code W-Langage, le compiler dynamiquement puis l'exécuter en gérant les exceptions.
- ▶▶ Crypter des chaînes de caractères et des fichiers externes en mode sécurisé, selon un algorithme 128 bits.
- ▶▶ Concevoir un éditeur de requêtes interactives, générant des interrogations en langage SQL et les utiliser au sein de l'application.
- ▶▶ Implémenter un proxy de messagerie POP/SMTP.

### Pré-requis

*Pour valider les pré-requis nécessaires, avant d'aborder le TP, indiquez si les affirmations suivantes sont vraies ou fausses :*

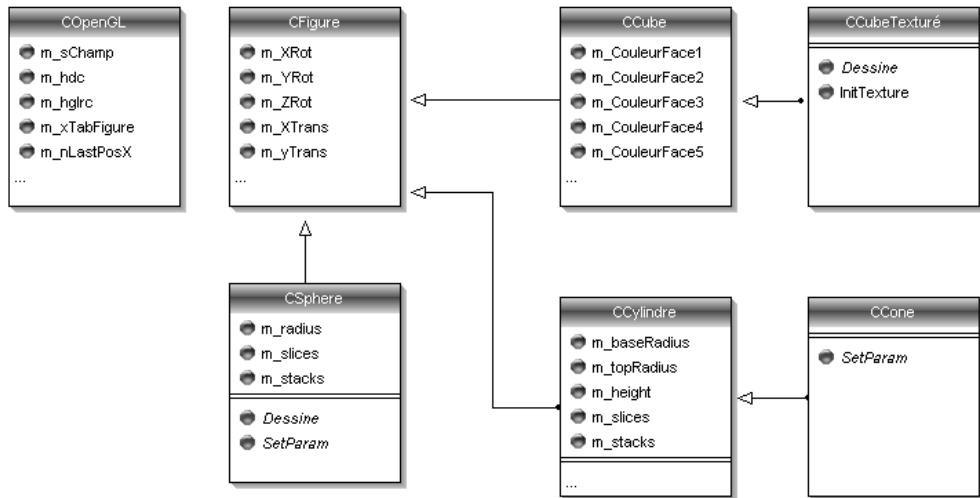
1. UML permet de représenter visuellement une hiérarchie de classes, par rétro analyse du code source.
2. Les paramètres passés à une méthode de classe peuvent être typés explicitement ou non.
3. Le document de définition d'un fichier XML dépend de la plate-forme sur laquelle les données vont être exploitées.

4. Une procédure générée dynamiquement doit être déclarée en EXTERNE pour que n'apparaisse pas de "warning".
5. Un fichier Hyper File crypté ne peut pas être relu sans fournir à nouveau la clé de protection qui a permis de le générer.
6. Une requête dont le code SQL est créé par programme doit être exécutée avec le paramètre `hRequeteSansCorrection`.

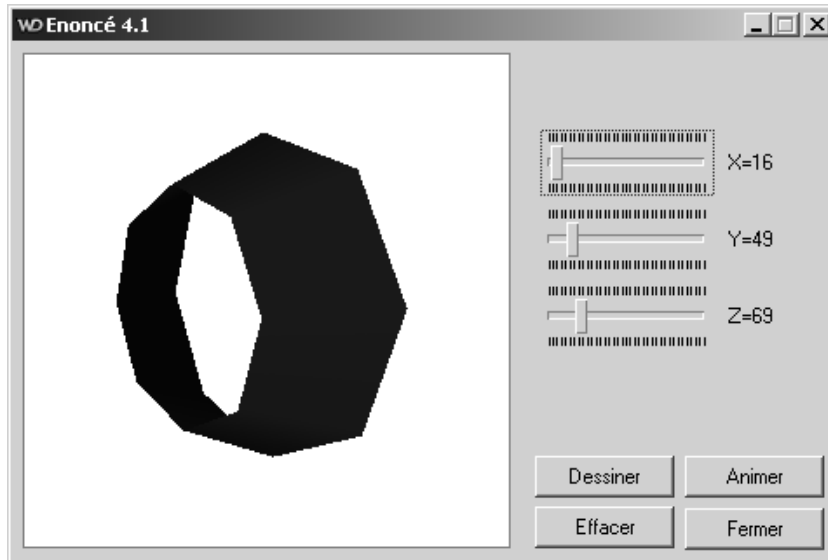
## Énoncé 4.1 : Programmation Orientée Objet

**Durée estimative :** 1 h 20 minutes

Ce TP permet d'explorer les possibilités du langage WinDev (W-Langage) en matière de programmation orientée objet : conception et utilisation de classes, hiérarchisation des objets (classes ancêtres, classes dérivées) et leur représentation visuelle sous forme de diagramme UML :



L'exercice ci-après se propose de tester les capacités de modélisation OpenGL au travers de classes pilotant les APIs de Windows. Il s'agira ici de dessiner un objet en 3D et de le manipuler à la souris et par programme : déplacement, zoom, rotation, animation, etc.



1. Pour commencer, nous allons présenter à l'utilisateur une fenêtre WinDev contenant un objet dessiné en 3D à l'aide d'OpenGL. Pour cela, créez une fenêtre avec un champ image, puis importez les classes C\* .WDC fournies avec l'exemple didactique **WD OpenGL**.
2. Écrivez le traitement permettant d'initialiser le module OpenGL, en lui associant le champ image créé précédemment. Observez les méthodes de la classe `cFigure`. Que remarquez-vous concernant certaines procédures ?
3. Quand l'utilisateur clique sur le bouton **Dessiner**, l'objet qui doit être représenté est un cylindre bleu à 8 faces, dont la base a un rayon de coefficient 3 et un rayon haut de coefficient 2. Écrivez le code adéquat et testez la fenêtre pour vérifier que vous obtenez bien la figure attendue.
4. Pour comprendre les imbrications des différents objets, construisez le modèle UML du projet par rétro-analyse. Faites évoluer votre objet en forme de cône. Comment les classes W-Langage pourraient évoluer en partant du modèle UML (opération inverse) ?
5. L'objet doit apparaître en 2 dimensions initialement (vu du dessus), puis en 3 dimensions lorsqu'il est déplacé à la souris. Modifiez votre programme pour qu'il apparaisse directement en 3D et redéfinissez sa couleur. Effectuez des essais puis observez comment les constantes de couleurs sont décodées avant d'être transmises aux APIs.

6. Quand l'utilisateur clique sur le bouton **Effacer**, l'objet doit disparaître. Écrivez le code pour supprimer la figure et réinitialiser le zoom de la caméra.
7. L'orientation de l'objet dans l'espace doit être paramétrable par l'utilisateur dans chacun des trois plans (largeur, hauteur, profondeur). Pour cela, vous allez créer trois potentiomètres linéaires permettant de régler les paramètres de rotation X, Y, Z. Redessinez l'objet à chaque changement de position d'un des potentiomètres.
8. Quand l'utilisateur clique sur le bouton **Animer**, l'objet doit tourner sur lui-même dans l'espace, de gauche à droite et de haut en bas, comme un projecteur. Faites varier l'orientation à l'infini, sur 360 degrés selon chaque axe de rotation. La vitesse de rotation sera de 10 degrés tous les dixièmes de seconde.
9. Quand l'utilisateur appuie sur (Echap), l'animation doit s'arrêter. Modifiez le code du bouton **Animer** en conséquence.

## Indices pour l'énoncé 4.1

*Vous pouvez vous inspirer de l'exemple **WD OpenGL** pour maîtriser les opérations les plus courantes : représentation de cubes, de sphères et de cylindres.*

*Écartez de votre diagramme UML les classes RAD, qui ne font pas l'objet du présent TP et n'interviennent pas dans l'exercice.*

*Pour appeler la méthode `SetRotation`, définissez des variables de type réels car les paramètres sont typés explicitement.*

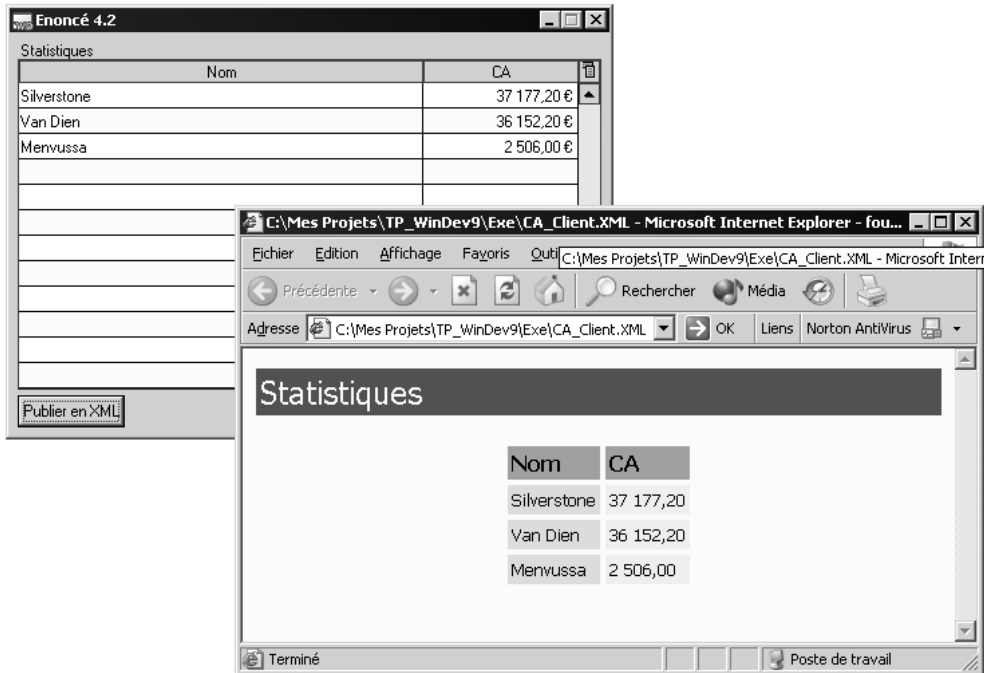
*Une temporisation est indispensable pour l'animation afin de contrôler la vitesse de rotation d'une part, et de permettre une sortie du programme d'autre part (détection de la touche (Echap)).*

## Énoncé 4.2 : Utilisation de XML

**Durée estimative** : 45 minutes

La connaissance du langage XML est un atout permettant au développeur de normaliser ses échanges de données entre plates-formes hétérogènes. Peu importe le langage de programmation utilisé pour générer le fichier (ou la trame) XML, les données restent exploitables. Il en va de même pour les systèmes d'exploitations. L'exercice ci-après utilise les fonctions XML du W-Langage pour lire et écrire des tableaux de données, en vue de les échanger (via Internet par exemple).

Le TP suivant propose de générer un fichier XML d'après les données contenues dans une table WinDev. Les fonctions de relecture et de mise à jour sont également étudiées. Dans une fenêtre, vous allez créer une table permettant de visualiser les statistiques de Chiffre d'Affaires et de les publier ensuite en XML (dans un Intranet par exemple) :



1. Quand la fenêtre s'affiche, le chiffre d'affaires des 10 meilleurs clients doit apparaître sous forme de tableau. Utilisez une requête pour extraire les données et les afficher dans une table.
2. Quand l'utilisateur clique sur le bouton **Publier en XML**, créez un fichier XML avec les données de la table.  
Que constatez-vous concernant le formatage des numériques ? Comment y remédier ?
3. Les montants affichés dans le document XML doivent être formatés avec un séparateur de millier (espace), un séparateur décimal (virgule) et présentés avec deux décimales. Pour cela, procédez à la relecture du fichier XML généré précédemment.

4. Chaque chiffre d'affaires présent dans le fichier doit être retrouvé parmi les balises du document. Écrivez le traitement permettant de parcourir l'arborescence XML depuis la racine puis d'extraire uniquement les montants voulus (balises **CA**). Tracez pour vérifier.
5. Procédez au formatage de chaque élément et mettez à jour le document XML. Reconstituez le code source incluant les balises XML et réécrivez le fichier sur le disque. Vérifiez en éditant le fichier résultat.
6. L'utilisateur doit voir apparaître automatiquement le résultat sous Internet Explorer. Rajoutez le code correspondant.  
Sauriez-vous publier ces données au sein d'un Intranet ? Comment ?

## Indices pour l'énoncé 4.2

*Se reporter à l'exercice du TP 2.3 pour la première partie de l'exercice (extraction des données à l'aide de la requête).*

*Le contenu d'une table peut être exporté en XML en une seule opération. Le parcours ligne à ligne peut être évité.*

*Pour la relecture, le contenu du fichier sur disque doit être chargé au préalable en mémoire pour créer le document XML.*

*Les données lues dans le document XML sont renvoyées sous forme de chaînes ; il est donc nécessaire de les convertir en numérique dans le cadre de cet exercice.*

## Énoncé 4.3 : Compilation dynamique

**Durée estimative** : 40 minutes

Cet exercice aborde la génération du code W-Langage, à la volée, juste avant de l'exécuter. Ce procédé est appelé compilation dynamique. Le code source est stocké sous forme de chaînes de caractères, puis compilé sous forme d'une procédure, qui peut être immédiatement appelée. Cette méthodologie permet notamment d'offrir à l'utilisateur la possibilité de saisir lui-même des formules de calcul paramétrables.