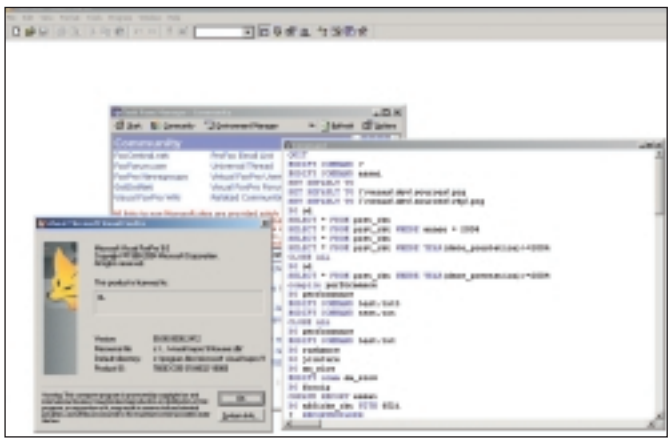
Visual FoxPro Europa : la meilleure version depuis Visual FoxPro 3

Europa est le nom de code de la dernière mouture de Visual FoxPro proposée par Microsoft. Cette version 9 apporte de nombreux changements, comme un nouveau générateur de rapports ultra puissant supportant GDI+.

Les principales caractéristiques de VFP

La caractéristique principale de Visual Foxpro est son interface de développement. Si vous lancez le logiciel vous vous retrouvez en mode ligne de commande avec, à votre disposition, un moteur de base de données embarqué. VFP est à la base un interpréteur et cette approche permet de développer des applications très rapidement. Il s'agit à notre avis d'un des meilleurs outils de développement pour petites entreprises voulant élaborer des applications verticales de gestion (comme l'encodage de prestations, un module de facturation, la génération de rapports divers, la création de fiches de paies, etc.). VFP 9 (build 2412) n'est pas intégré à Visual Studio 2005 et c'est une bonne chose : en effet le runtime d'EUROPA occupe moins de 5 Mo et une installation de base ne nécessite pas 220 Mo d'espace disque. Nous sommes loin des besoins gargantuesques de Visual Studio ! Cependant, l'interaction entre Visual Studio et VFP a été améliorée : amélioration d'une part du support XML qui existait déjà avec VFP 7 et VFP 8 (encadré 1), mais aussi introduction de nouveaux types de données pour une meilleure compatibilité avec Microsoft SQL Server 2000 et SQL Server 2005 (Varchar, Varbinary et Blob).



Système d'exploitation : Windows XP recommandé, exécution sous Linux via l'émulateur Wine possible
Langages : SQL et langage propre à Visual Foxpro
Besoins : 250 Mo d'espace disque, 256 Mo de RAM, machine cadencée à au moins 1 Ghz

Encadré 1

Sous VFP le XML est omniprésent. Vous pouvez convertir une table en un format XML et inversement :

```
CLOSE DATABASE ALL
USE HOME(2)+"\data\customer"
CURSORTOXML("customer","lcXML",1,32)
STRTOFILE(STRCONV(lcXML,9),"customer.xml")
XMLTOCURSOR("customer.xml","curCustomer",512)
RETURN
```

On peut aussi associer à chaque fichier XML un schéma (XML Schema Definition (XSD)), et ainsi créer ou manipuler celui-ci par programmation (à travers l'API XML Schema Object Model (SOM)). Ce qui donne ceci, par exemple, pour créer un curseur à partir d'un schéma :

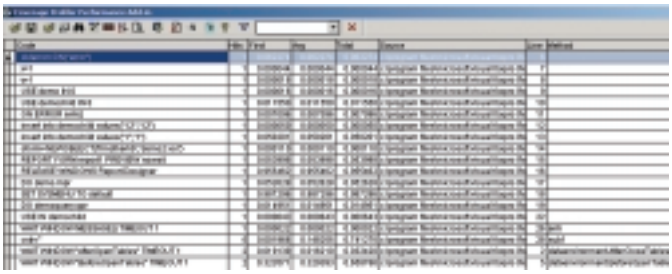
```
adapter = CREATEOBJECT("XMLAdapter")
cFile = "Votre_Chemin\StructureXML.xml"
adapter.LoadXML(cFile,.T.)
adapter.Tables(1).Fields(1).DataType = "C"
adapter.Tables(1).Fields(1).MaxLength = 10
adapter.Tables(1).ToCursor()
```

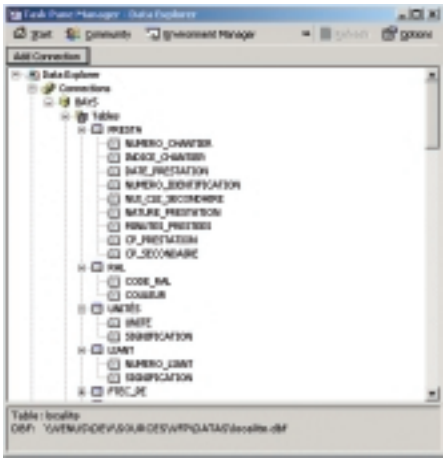
1 Les améliorations de l'IDE

Chaque nouvelle version de VFP a toujours été accompagnée d'amélioration de l'IDE et cette version 9 ne fait pas exception à cette règle.

L'IDE de VFP est très comparable à celui de VB. On y retrouve la coloration de syntaxe, l'intellisense, y compris sur les objets COM, des fenêtres ancrables, un gestionnaire de projet arborescent, un débogueur avec trace, call, watch, locals, points d'arrêts par click, une liste des tâches, une intégration avec Visual Source Safe, la mémorisation de la taille, de la position, et de la présentation des fenêtres. L'intellisense y est adaptable (introduction de vos propres fonctions et procédures à l'aide de l'intellisense manager), la fenêtre Document View présente un sommaire complet des modules existants dans un fichier programme, une classe ou un écran. Enfin, il y a la fenêtre des commandes qui a forgé une grande partie de la réputation de Foxpro. Dans celle-ci, vous pouvez y exécuter toute commande ou groupe de commandes pour test avant son intégration dans un programme.

Avec cette nouvelle version, vous pouvez laisser le soin à l'IDE de compiler votre code à l'arrière plan (il suffit d'activer la coloration syntaxique), ce qui est vraiment un plus concernant la détection d'erreurs. Pour l'optimisation du code, un outil de profilage a été ajouté (add-in coverage profiler).





En outre, VFP 9 est livré avec un nouvel outil le DATA EXPLORER que l'on peut activer du Task Pane Manager. Il sera utilisé pour parcourir des tables de bases de données locales Foxpro ou externes (comme SQL Server), créer des requêtes et en visualiser le résultat, etc.

2 Le SQL revu et corrigé

Avant VFP 9, la syntaxe du SQL de Foxpro datait de la version 2.6 pour DOS... Ce dialecte SQL a toujours été non compatible avec le standard ANSI. VFP 9 améliore enfin ce point crucial, tout en conservant une compatibilité avec l'existant si nécessaire. La syntaxe proposée est maintenant fort proche des produits SQL Server de Microsoft. Une table VFP peut comporter jusqu'à 255 champs et 1 milliard d'enregistrements. Chaque champ peut revêtir un des 12 types correspondants aux standards SQL. On peut associer à une table un fichier index comportant des clés en nombre illimité.

Malheureusement, la limite de 2 gigas pour une table n'a pas été franchie. C'est dommage, mais compréhensible dans le sens où intrinsèquement VFP 9 est un produit 32 bits (il pourra tourner sous une architecture 64 bits mais en mode compatibilité 32 bits). Et puis, et surtout, Microsoft ne désire pas empiéter sur son moteur de base de données haut de gamme SQL Server (remarquez que le produit, pourtant de base, SQL Server 2005 Express Edition, peut atteindre la limite de 4 gigas par table !). Ceci dit, dans d'autres domaines certaines limites n'existent plus, comme le nombre de jointures et de sous-requêtes, ou le nombre d'UNIONs au sein d'un SELECT. Une requête peut faire référence à un nombre illimité de tables, et vous pouvez avoir plus de 24 arguments (note 1) dans un SQL IN (select * where variable in (1,2,3,4,...,25,26,27,...)). Les performances ont été améliorées dans plusieurs domaines. Nous avons réalisé des tests. Au niveau de la vitesse d'interprétation pure VFP 9 est plus lent que ses prédécesseurs. Cette constatation peut s'expliquer : plus de mots clés (tokens) sont reconnus par la boucle d'interprétation. Au niveau des requêtes SQL, il y a par contre une nette amélioration si on tient compte de ce qui a été optimisé, comme l'utilisation des mots clés LIKE ou TOP dans un SELECT, ou encore de l'emploi du nouveau type d'index dit "binaire". Cet index encore appelé index bitmap, convient pour des index d'expressions logiques, comme par exemple "l'enregistrement est effacé ou non". Il tient beaucoup moins de place et sera par conséquent (beaucoup) plus rapide.

Enfin, VFP 9 permet de créer une application à partir d'une base de données VFP et, si nécessaire, la transférer sur une base SQL sans changer le code de l'interface utilisateur.

Note 1

La limite du nombre d'arguments est fixée par un appel à SYS(3055)
SYS(3055 [, nComplexity])
Par défaut ce niveau de complexité est fixé à 320 mais peut varier jusqu'à 2040.

3 Un gestionnaire de rapports plus élaboré

Ici aussi les concepteurs ont décidé de ne pas entraver la compatibilité descendante : vous pouvez toujours éditer ou utiliser vos rapports conçus avec une version antérieure : c'est à vous de décider de basculer ou non d'un générateur à l'autre.

```
*– Utilise le nouveau générateur
_REPORTBUILDER = HOME() + 'ReportBuilder.app'

*– Utilise l'ancien générateur
_REPORTBUILDER = ''
```

Le nouveau Report Writer peut maintenant partager avec d'autres rapports un environnement de données. Et si vous le désirez, cet environnement de données (bases de données, tables et curseurs qui sont employés par le générateur), sera à son tour sauvegardé en une classe pouvant être rechargée ultérieurement. Le gain de temps est appréciable, d'autant plus que vous pouvez, de la même manière, créer un rapport en vous basant sur l'environnement de données d'un autre. Nouvelle fonction : chaque objet défini au sein de votre layout pourra être protégé : par exemple, tel champ ne pourra être déplacé, ou effacé, ou sera bloqué dans ses proportions sur le layout. Cet aspect des choses est important pour développer rapidement, car on n'est jamais à l'abri d'une mauvaise manipulation qui effacerait ou changerait de position un groupe de champs (le rapport complet pourra être protégé ; ou vous déciderez de ne protéger que des éléments individuels).

Un des plus grands changements se situe dans la création de bandes de détails multiples. Cette possibilité assez révolutionnaire pour VFP, autorise un développeur à traiter plusieurs tables enfants pour un enregistrement parent.

```
*– Ouvertures des tables enfants
USE Membres IN 0 ORDER ClientsClePrimaire
USE Vehicules IN 0 ORDER ClientsClePrimaire
USE Habitations IN 0 ORDER ClientsClePrimaire

*– Ouverture de la table parent
SELECT 0
USE clients ORDER ClientsClePrimaire

*– Mise en place des relations
```

```
SET RELATION TO ClientsClePrimaire INTO Membres
SET RELATION TO ClientsClePrimaire INTO Vehicules ADDITIVE
SET RELATION TO ClientsClePrimaire INTO Habitations ADDITIVE

*- Génère le rapport
REPORT FORM ASSURANCE PREVIEW
```

Ainsi pour chaque client vous pouvez obtenir plusieurs bandes détails :

[Première bande détail]

Client : Monsieur Jean DUPONT
Rue du paradis

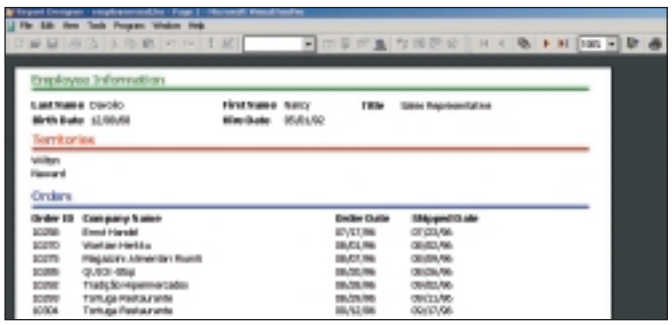
Membres :	Nom	Date naissance	
	Jean Dupont	01/02/1963	
	Céline Dupont	28/03/1965	
Véhicules :	Année	Type	Valeur
	1983	Ford	xx Euros
	1990	Opel	xx Euros
	1995	Peugot	xx Euros
Habitation			
	Année construction		
	1954	Rue du paradis	xx Euros

[Deuxième bande détail]

Client : Monsieur Charles Petit
Rue de l'enfer

Membres :	Nom	Date naissance	
	Charles Petit	01/04/1958	
Véhicules :	Année	Type	Valeur
	2000	Ford	xx Euros
	2001	Renaud	xx Euros
Habitation			
	Année construction		
	1978	Rue de la résistance	xx Euros
	1985	Rue de l'enfer	xx Euros

etc.



4 VFP 9 et .NET

Il est facile de créer un service Web avec Visual Studio puis de le consommer sous VFP 9. Le code .Net en C# :

```
[WebMethod(Description="Un exemple simpliste")]
public string Bonjour(string Nom)
{
   return "Bonjour, " + Nom + "! Il est maintenant : " +
          DateTime.Now.ToString();
}
```

Le code VFP 9 :

```
o = CREATEOBJECT("MSSoap.SoapClient30")
? o.MSSoapInit("http://localhost/VFPWebService/" +
          "VFPInteropService.asmx?WSDL")
? o.Bonjour("Xavier")
```

Le contraire est aussi envisageable, quoique plus compliqué : vous pouvez créer une application Web avec ASP.NET qui interrogera un service Web construit en VFP. Par exemple vous pouvez interroger à distance votre base de données Foxpro, et ce mécanisme d'échange retournera le résultat de vos requêtes SQL. Remarquez que vous pouvez déjà interroger vos tables DBF en ASP.NET via OLEDB, comme le montre cet extrait de code :

```
...
private bool GetCustomerList(string lcCursor)
    {
      this.oConn = new OleDbConnection("Provider=vfpoledb.1;" +
        @"Data Source=C:\Programs\vfp\Samples\Data\testdata.dbc;" +
         "Exclusive=false;Nulls=false");
      try {
        this.oConn.Open();
      }
      catch(Exception ex)
      {
        this.lblErrorMsg.Text = ex.Message;
        return false;
      }
      string lcSQL = "select company, contact, city from customer";
    OleDbDataAdapter oAdapter = new OleDbDataAdapter(lcSQL,this.oConn);
      if (this.oDS == null)
      {
        this.oDS = new DataSet();
      }
      try
      {
        oAdapter.Fill(this.oDS,lcCursor);
      }
      catch(Exception ex)
      {
        this.lblErrorMsg.Text = ex.Message;
        return false;
      }
      finally
      {
        this.oConn.Close();
      }
      return true;
    }
...
```

Enfin, vous pouvez employer des composants .Net en VFP 9 via COM, ou automatiser des applications depuis VFP 9 via COM.

```
...
* – Extrait de code de l'automation d'Open Office
PUBLIC goOOoDesktop
goOOoDesktop = OOoServiceManager_CreateInstance( "com.sun.star.
frame.Desktop" )
COMARRAY( goOOoDesktop, 10 )
...
```

5 Migrer plus facilement vers .NET avec l'assemblage VFPtoolkitNET

Depuis 2002, il existe un assemblage du nom de VFPtoolkitNET.DLL qui comprend plus de 250 fonctions FoxPro, permettant à un développeur .Net d'utiliser en code managé des fonctions VFP comme CreateObject() ou DOW(). Ce qui donne par exemple en C# :

```
DateTime tDateTime = DateTime.Now;
int nDow = VFPToolkit.dates.DOW(tDateTime);
```

Cet assemblage a été placé dans le domaine public. Son code source est lui aussi disponible (l'implémentation des fonctions VFP, soit en Visual Basic.NET soit en C#). Ainsi, l'implémentation de la fonction (simple) DOW en C# est la suivante : public static int DOW(System.DateTime dDate) {return (int)dDate.DayOfWeek;}

6 Faire appel à GDI+

Nous avons gardé le meilleur pour la fin. Microsoft Visual FoxPro 9 utilise GDI+ en interne. Avec les précédentes versions de VFP, pour afficher un graphique dans un formulaire vous deviez obtenir au préalable un "contexte" via les API Windows, et cette possibilité était complètement masquée avec le générateur de rapports. Avec VFP 9 est livrée une nouvelle propriété GDIPlusGraphics de la classe ReportListener, qui contient un handle utilisable par votre code VFP.

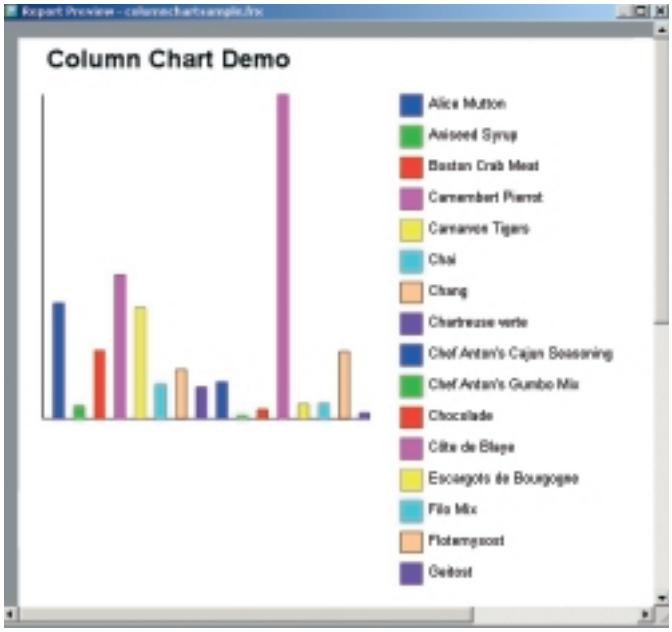
```
...
if not .IsSuccessor
        .SharedGDIPlusGraphics = .GDIPlusGraphics
    endif not .IsSuccessor
    .oGDIGraphics.SetHandle(.SharedGDIPlusGraphics)
...
```

En clair, vous pouvez dorénavant afficher par programmation un graphique dans vos rapports ! Les possibilités sont ici fort nombreuses.

EXEMPLE

Affiche la liste des polices de caractères disponibles dans un rapport :

```
CREATE CURSOR crsrFontList ( ;
        cFont char(32))

llGotFonts = AFONT(laFonts)
FOR i = 1 TO ALEN(laFonts)
        INSERT INTO crsrFontList VALUES (laFonts(i))
ENDFOR

CREATE REPORT fontList FROM DBF("crsrFontList")
loListen = CREATEOBJECT("FontListener")

REPORT FORM fontList PREVIEW OBJECT loListen

DEFINE CLASS FontListener as ReportListener
ListenerType = 1

PROCEDURE EvaluateContents
        LPARAMETERS nFrxRecno, oObjProperties

        oObjProperties.FontName = ALLTRIM(cFont)
        oObjProperties.Reload = .T.
ENDPROC
```



Pour conclure

En ce qui concerne l'avenir de Foxpro il serait facile de déclarer "il n'y a actuellement aucune assurance pour que VFP 10 soit un jour publié, le marketing de Microsoft poussant essentiellement la branche SQL Server et en particulier les derniers nés de la famille Express". C'est juste, mais cette même litanie se répète depuis Visual Foxpro 3 !

De l'aveu même de l'éditeur 100 000 développeurs utiliseraient encore VFP dans le monde, essentiellement pour le compte de petites entreprises. Il est clair que ceux-ci ont eu des années pour changer d'outil. Et s'ils ne l'ont pas fait c'est qu'il existe de très bonnes raisons pour cela. VFP n'est pas gratuit, mais il est très abordable. En outre les programmes développés avec VFP (exécutables) sont distribuables avec le runtime VFP gratuitement. En plus d'être abordables, tout est compris dedans : un générateur de report, une panoplie d'assistants, une puissante base de données et un runtime. VFP 9 nous permet aujourd'hui de nous connecter à toutes les bases de données du marché par oledb, odbc, xml, etc. VFP n'est pas Access, ni SQL Server. Il s'agit bel et bien d'un outil unique en son genre pour développer très rapidement une application de gestion. Microsoft nous livre ici un excellent cru. L'essayer c'est l'adopter.

Xavier Leclercq - *xavier.Leclercq@programmez.com*