Petite notice explicative

Le tableau « arbres.pdf » reprend l'ensemble des controls, méthodes et propriétés des objets dont je n'ai pu achever la description lors de ma session lyonnaise.

Il est organisé en cinq colonnes qui correspondent, pour chacune d'entre elle, à l'un des centres d'intérêt exploré lors de cette démonstration.

Evénements.

C'est une procédure classique de trace texte de certains événements. Il s'active en positionnant à .T. la propriété lTraceFile de vfpForm ou en affectant une valeur alphanumérique à la variable globale gcTraceFile.

Objets.

De nombreux objets sont créés et détruits lors de la manipulation des classes que je vous propose. On pourrait laisser faire le garbage collector. Pour des raisons de performance, j'ai préféré gérer « à la main » le cycle de vie des objets (voir ma session de Roissy en 2013).

Afin de contrôler que tout ce passe bien, ou pourquoi cela ne fonctionne pas, les différents cycles de la vie d'un objet sont mémorisés dans gaObjects de type vfpArrayList.

Les événements tracés sont : Load(), Init(), QueryUnload(), onRelease(), Destroy(), UnLoad().

Démonstration.

Comme vous avez pu le constater, il est parfois utile de ralentir l'exécution d'une procédure soit même de la suivre pas à pas. Les éléments surlignés dans cette colonne participent de cette fonctionnalité.

Localisation.

Quelques propriétés afin d'assurer la portabilité du code.

cHomeDir est automatiquement affecté.

cTestDir est le répertoire qui héberge les fichiers de test (TXT et DBF).

cTmpDir est le répertoire dans lequel seront stockés les fichiers de trace.

cImgDir est le répertoire hébergeant les images nécessaires en phase de développement.

Observer.

Les objets manipulés au cours de cette session communiquent fortement entre eux. J'ai pour cela implémenter une version allégée du pattern « Observer » présenter dans ma session de Roissy en 2015.

J'ai également, et par souci de clarté et de simplicité, utilisé le rapport parent-enfant du formulaire et des objets qu'il contient : appel de l'intérieur d'un objet de THISFORM, appel depuis le formulaire de la méthode d'un objet.

J'ai également, dans la partie création de l'arbre et décorticage des rotations, sous-classé la classe vfpTree afin d'y intégrer le code de pas à pas.

Comme je vous l'avais indiqué en 2015 (Roissy) et 2016 (Marseille) la communication entre objets est un vaste sujet qui nécessite tact et analyse minutieuse. Respecter de trop près le schéma du design pattern peut facilement nous faire sombrer dans la complexité inutile, voire le bouclage infini de procédures s'auto-appelant.

Je ne suis pas sûr d'avoir gagné le pari. Beaucoup reste à faire pour assurer solidité, efficacité et lisibilité

Et les autres objets ?

Vous avez pu les observer lors de la session de cette année. Si nécessaire, je peux rédiger une contribution particulière sur la gestion des arbres.

Concernant les arbres n-aires, que du grand classique avec l'objet OLE TreeView proposé par VFP.

Pour les arbres binaires, je n'ai rien trouvé dans l'environnement VFP – ni dans les autres environnements – pour visualiser simplement une telle structure.

Ce que je vous propose est le fruit d'un travail non nécessairement très volumineux mais qui a exigé de ma part imagination et persévérance. Si certains, parmi vous, pensent que le travail mérite d'être poursuivi, je suis disposé à le faire en compagnie des volontaires intéressés par un tel projet. Si vous utilisez le travail présenté à la fin de la session sur les arbres binaires, je vous demande d'avoir l'amabilité de bien vouloir me citer.

Paris le 21 mars 2018

Marc Thivolle