



ATOUTFOX

Session 15-03-2018 10h45

VFP sur le WEB

Démarche et exemple simple **d'adaptation d'un petit projet VFP** **sur le WEB avec FIC**

1- Présentation de l'outil FoxInCloud de la société **Abaque**

- a - A quoi sert FoxInCloud ? : c'est un puissant outils d'adaptation de la couche visuelle des méthodes et événements d'objets Visual Foxpro 9-sp2, pour être affichés et simulés dans un navigateur internet

- b - Sur quelles technologies s'appuie-il? F.I.C (FoxInCloud) utilise :
 - coté serveur WEB Microsoft IIS8.0 et les ressources de West-Wind Web Connect de Rick Strahl couplées à la technologie Ajax pour les interactions avec les navigateurs clients
 - coté client, le langage Javascript et les bibliothèques JQuery, BootStrap, ActiveWidgets ainsi que les normes W3C HTML5 / CSS3 pour l'affichage dans les navigateurs

2- Présentation du projet d'exemple **ATDFDEMO**

- a- ATDFDEMO permet à un utilisateur de créer des demandes de réparation au titre d'un SAV
- b- Que contient ce projet:
 - deux formulaires l'un pour s'identifier (login et mp), l'autre pour saisir une demande de réparation
 - une classe VCX dérivée de la classe de base de VFP
 - trois PRG, un de lancement, un d'initialisation du contexte et un contenant des procédures et fonctions diverses utiles à l'application
 - un dbc contenant une connexion à une base PostGreSQL à travers ODBC, des vues paramétrées sur les tables
 - un fichier ATDFDEMO.SQL de création des tables dans PostGreSQL à l'aide de PGADMIN

3- Adaptation complète projet ATFDEMO

Cela s'effectue à travers sept phases principales, De la phase A à la phase F

A - F.A.A Analyse

Cette phase permet à F.A.A (l'assistant d'adaptation de F.I.C) de préparer l'ensemble de ses fichiers de travail et de recenser les tâches d'adaptation à réaliser

- 1 - Lancement de F.I.C affichage de son menu spécifique
- 2 - ouverture projet ATFDEMO
- 3 - lancement du projet sous IDE et test
- 4 - Lancement de l'assistant d'adaptation et commentaire sur l'écran F.A.A et ses fonctions
- 5 - Sélection et exécution de l'analyse sur code source

B - F.A.A Adaptation

Cette phase crée une bibliothèque de classes dérivée de celle de F.I.C, remplace les classes natives par les sous-classes de F.I.C, les formulaires et les objets, adapte en modifiant les codes sources des bibliothèques de classes, des fichiers de procédures ou fonctions, des formulaires, des objets et des menus et enfin recense, indique et commente les adaptations manuelles à réaliser

- 6 - Exclusion des PRG non graphique (utile.prg) et lancement de l'adaptation

#Il est recommandé d'exclure des fichiers PRG ne contenant aucune procédure ni fonction intervenant sur la partie graphique pour éviter un travail inutile pour F.A.A (gain de temps, suppression de risque d'erreur d'interprétation du code par F.A.A, etc..)

- 7 - Sélection sous F.A.A du filtre d'adaptation manuelle après adaptation étape 2. *Adaptations manuelles du code source*

Un certain nombres de commandes, d'événements et méthodes d'objets VFP ne peuvent être automatiquement traités par F.A.A.

C'est le cas pour la gestion:

- *du chargement des vues requery()*
- *des méthodes (callback) de traitement en retour d'appel d'un formulaire enfant (IF messagebox = myValue ELSE ...THEN, DO FORM myForm TO myVariable)*
- *des événements de grille BeforeRowColChange() et AfterRowColChange() qui doivent être séparés en traitements pour chaque nature de déplacement (colonne ou ligne)*
- *des commandes interdites dans un fonctionnement en mode WEB (cancel, quit, etc..) (l'application serveur ne doit pas s'arrêter autrement que par l'arrêt demandé par le service WEB de IIS)*
-

- 8 - Sortie de F.A.A, reprise depuis F.I.C des adaptations manuelles, et propriétés sauvegardées. Test dans l'IDE VFP du projet adapté

Les propriétés sauvegardées des objets sont, pour le bon fonctionnement de F.I.C, obligatoirement à renseigner dans la propriété wcPropSave de chaque objet.

Lors du traitement d'une requête HTTP, à chaque changement par programmation d'une propriété d'un objet (exemple la couleur de fond BackColor d'un objet textbox qui varie selon la valeur saisie), F.I.C doit rétablir les propriétés de chaque objet de la page. Il faut donc, au préalable, qu'il sauve après le traitement d'une précédente requête HTTP toutes les propriétés de tous les objets d'une page HTML pour les restituer correctement au traitement de la prochaine requête HTTP.

Cette sauvegarde serait beaucoup trop longue à effectuer si F.I.C traitait toutes les propriétés de chacun des objets. Il est donc nécessaire de lui indiquer, par objet, quelles sont les propriétés à sauver / restituer limitant ainsi au minimum le temps de traitement.

Les propriétés à indiquer sont uniquement celles qui sont changées pendant l'exécution des méthodes / événements du formulaire ou des objets du formulaire.

C - F.A.A Publication

Cette phase publie l'application dans le serveur WEB IIS, génère les PRG et projets xxxTest et xxxProd contenant l'ossature F.I.C du projet final WEB

9 – Sous F.I.C lancement de la publication sous F.A.A, correction du PRG ATFSETS (pétouille F.I.C manque de remettre le ENDIF final), exécution ATFTEST.prg, ouverture navigateur localhost/atftest/, vérification de l'affichage des formulaires

La publication créé deux fichiers de lancement de l'application nommés xxxTEST et xxxPROD (le premier est utilisé pour un projet de test, le second est à utiliser pour le projet de production), un fichier xxxSets contenant les classes d'initialisation des services F.I.C et de l'application adaptée

D - F.I.C Projet WEB

Cette phase finalise le squelette F.I.C du projet adapté pour une exécution en mode WEB après compilation d'un exécutable (xxxTEST ou xxxPROD) à lancer en mode fichier ou en service DCOM

10 – Depuis F.I.C ouverture projet ATFTEST, ajout des PRG (atfdemo, atfinit, utile, atfSets), des bibliothèques de classes (gwsav, atf), des formulaires (index et sav), originaux adaptés du projet ATFDEMO. Régénération complète du projet pour faire remonter toutes les dépendances. Exclusion des PRG utilitaires de F.I.C (atpjclean, atpjcompile, atpjcompiledebugmode) et de lancement en mode desktop (atfdemo)

F.A.A prépare les deux squelettes de projets (xxxTEST, xxxPROD) mais n'y ajoute pas (dans sa version actuelle) automatiquement les fichiers sources du projet initialement adapté. Il faut ajouter manuellement dans le projet WEB (xxxTest, xxxProd) les fichiers adaptés du projet originel ATFDEMO

11 - Lancer ATFTEST pour vérifier le bon fonctionnement du serveur F.I.C adapté, lancer un navigateur en localhost/atftest/ et tester l'affichage des formulaires / pages HTML à l'aide de la page de test SampleIndex mise à disposition par F.A.A

12 - Faire d'index.scx la page par défaut en mode WEB. Ouvrir IIS

et ajouter dans l'option « programme par défaut » index.atf (nom du formulaire par défaut de l'application + extension choisi pour le projet), relancer IIS, modifier atfserver.prg pour remplacer dans la procédure INDEX sampleindex.scx par index.scx

E – F.I.C Finalisation

Cette phase permet de finaliser l'exécution de l'application en mode WEB en intégrant les sécurités d'accès aux pages et formulaires

Par défaut tous les formulaires adaptés par F.A.A sont indiqués comme n'étant accessible que par utilisateur authentifié par la propriété wAnonymousAllowed = .F. du formulaire. Si une page HTML / formulaire doit être accessible sans authentification préalable, alors la propriété wAnonymousAllowed du formulaire doit être à .T.

13 - Gérer les autorisations d'accès au page web / formulaire uniquement sur des utilisateurs authentifiés :

- Ouvrir le formulaire INDEX indiquer la propriété wAnonymousAllowed à .T. pour laisser son accès libre sans authentification préalable
- Ajouter dans les méthodes click() des boutons BTNConnect et BTNDisconnect l'appel à la méthode THISFORM.wUserLogIn() en y passant la clé d'identification de l'opérateur authentifié
- Ouvrir atfServer.prg et instruire les méthodes d'accès et de restitution de l'utilisateur authentifié. wUserGet() doit retourner la propriété _SCREEN._clepkUser contenant l'identifiant unique de l'utilisateur. wUserSet() doit appliquer à _SCREEN._clepkUser la valeur du paramètre m.twUser (attention à bien conserver le return dodefaut() dans cette fonction. wUserSet() est exécutée par F.I.C avant chaque traitement de requête HTTP pour repositionner l'utilisateur courant sur celui fourni par la requête, pour rétablir le contexte de cet utilisateur
- Exclure de la sauvegarde/restitution, par F.I.C, des propriétés de l'objet _SCREEN, la clé d'identification de l'utilisateur authentifié en indiquant _ScreenPropertiesSaveNot = '_clePkUser'

F.I.C doit gérer la sauvegarde et la restitution du statut authentifié ou non de chaque utilisateur. F.I.C s'en charge à travers les fonctions wUserSet() et wUserGet() du prg xxxServer. Ces deux fonctions sont exécutées lors du traitement de chaque requête HTTP. XxxServer.wUserGet() doit retourner l'identifiant unique de l'utilisateur fourni par l'application. Pour ce faire, il faut ajouter une propriété à l'objet _SCREEN qui contiendra la clé d'identification de l'utilisateur. (Par exemple, cette clé est stockée dans une table des utilisateurs de l'application et est retournée après la recherche par un login / mp à travers cette table). XxxServer.wUserSET(m.twUser) instruit dans l'objet _SCREEN la propriété contenant la clé d'identification avec le paramètre reçu twUser.

Lors de la méthode d'authentification dans le formulaire de connexion de l'application, il faut ajouter l'appel à la fonction THISFORM.wUserLogIn() en y passant l'identifiant unique trouvé par l'application pour cet utilisateur (par exemple l'ID trouvé dans la table des

utilisateurs) afin de fournir à F.I.C cet identifiant utilisateur. Chaque formulaire avec authentification utilisateur obligatoire vérifiera lors de son lancement que l'utilisateur courant est identifié et autorisé.
Pour que l'ensemble de ce processus fonctionne, il faut exclure du processus de sauvegarde / restitution des propriétés sauvegardées, la propriété du _SCREEN contenant l'identifiant de l'utilisateur authentifié en indiquant **dans le fichier xxxServer ScreenPropertiesSaveNot = 'le nom sans _SCREEN de la propriété à exclure'**

F - Déploiement de l'application WEB en mode EXE

Cette phase finale génère un exécutable autonome pour lancer l'application en mode WEB

14 - Lancer ATFTEST.prg et ouvrir le navigateur internet sur localhost/atftest/ pour vérifier le bon fonctionnement de l'application à travers l'IDE VFP / F.I.C

15 - Compiler un exécutable ATFTEST.EXE, lancer l'exécutable, ouvrir le navigateur internet sur localhost/atftest/ et vérifier le bon fonctionnement de l'application

Le déploiement le plus facile sur un serveur WEB de son application adapté sous F.I. pour moins de 5 utilisateurs connectés simultanément est de lancer directement l'exécutable. Lorsque le nombre d'utilisateurs simultanés dépasse les 5, alors il faut générer un exécutable pour un fonctionnement en service COM, paramétrer et enregistrer ce service COM sous Windows, paramétrer wConnect pour permettre un lancement de x sessions COM (x peut aller de 1 à 32). Ceci nécessite d'avoir les connaissances VFP et Windows pour générer et déployer des COM.

Ressources additionnelles

Blog:

<http://foxincloud.com/blog/>

FoxInCloud Marketing Videos:

<https://www.youtube.com/channel/UCUzzqO5375-fA7dLDds-wOQ>

FoxInCloud Technical Videos:

<https://www.youtube.com/channel/UCrEohAtbuf3uOXBH2pjp5Vw>

FoxInCloud Live Tutorial:

<http://foxincloud.com/tutoTest/>

FoxInCloud Live Tutorial (bootstrap):

<http://foxincloud.com/tutoTest/bs/>

FoxInCloud Demo:

<http://foxincloud.com/demoTest/>

FoxInCloud Demo (bootstrap):

<http://foxincloud.com/demoTest/bs/>

FoxInCloud Tastraders:

<http://foxincloud.com/tastrade/>

FoxInCloud Tastraders (bootstrap):

<http://foxincloud.com/tastrade/bs/>

Roadmap:

<http://foxincloud.com/roadmap.php>

Doc:

<http://foxincloud.com/how-to.php>

Téléchargement FoxInCloud Adaptation Assistant:

<http://foxincloud.com/download.php>

Téléchargement FoxInCloud Web Application Studio and Server:

<http://foxincloud.com/download.php#trialForm>