Interaction entre 4D 2003 et microsoft Excel (Services Web)

par Matthieu Chevrier

Date de publication : Février 2003

Dernière mise à jour :

Permettre à différentes applications d'interagir au travers de quelques protocoles répandus et standardisés (principalement SOAP et WSDL), telle est la passionnante avancée technologique proposée par les services Web. 4ème Dimension 2003 rend très simple la publication ou l'appel des services Web. Dans cette note technique, nous nous contenterons d'aborder les possibilités de publication (côté serveur) de 4D.

I - Introduction

Permettre à différentes applications d'interagir au travers de quelques protocoles répandus et standardisés (principalement SOAP et WSDL), telle est la passionnante avancée technologique proposée par les services Web.

4ème Dimension 2003 rend très simple la publication ou l'appel des services Web. Dans cette note technique, nous nous contenterons d'aborder les possibilités de publication (côté serveur) de 4D.

Depuis sa version 2000, Excel peut se transformer en client de service Web grâce à VBA (*Visual Basic Application*) et au *Microsoft SOAP toolkit*.

Le défi à relever se présente donc en ces termes : comment réaliser un connecteur Excel/4D à l'aide des Services Web ?

II - Objectifs

Afin d'écrire un Add-In Excel fournissant l'accès direct à une base 4D facilement depuis Excel, il nous suffira d'ajouter puis de publier une méthode dans une base 4D existante.

La conception d'un connecteur Excel/4D complètement générique est au-delà du périmètre de cette note technique. Cependant, vous pouvez facilement réutiliser et adapter les techniques décrites et le code fourni afin de satisfaire vos propres besoins.

Un autre objectif de cette note technique consiste à montrer une application pratique des services Web. Les solutions à base de services Web se développent facilement, et vous trouverez de multiples idées d'application à vos domaines d'activité.

III - Pré-requis

4ème Dimension 2003 avec activation des fonctionnalités de publication Web (vous pouvez utiliser le mode démo du serveur Web pour vous faire une première idée) - PC ou Mac ;

Excel 2000 et VBA (*Visual Basic Application*) - version Windows seulement (à la date de cette rédaction Excel sur OSX ne semble pas disposer de fonctionnalité SOAP) ;

Le *Microsoft SOAP* toolkit (version 3.0 minimum) - disponible gratuitement en téléchargement sur le site Web de Microsoft.

IV - Un peu de vocabulaire

Commençons par présenter quelques termes importants des services Web :

RPC : (*Remote Procedure Call*) Appel de Procédure à Distance ; il s'agit de la capacité d'une application à exécuter du code à la demande d'une autre application (lui transmettant des paramètres, recevant des résultats en retour) au travers d'un réseau.

HTTP : (*Hyper Text Transfer Protocol*) Protocole de transfert hypertexte ; constitue le protocole retenu pour la navigation sur Internet (demande de fichiers, envoi d'information, etc.).

Service Web : une collection de fonctions réunies dans une seule entité publiée sur un réseau.

SOAP : (*Simple Object Access Protocole*) : c'est le protocole utilisé par les services Web pour effectuer du RPC. Il s'agit d'un standard largement reconnu par la communauté des éditeurs de logiciels. SOAP se fonde sur XML, sans préjuger de l'emploi d'un protocole de transport particulier. Autrement dit, la spécification SOAP ne précise pas comment transporter les données au travers du réseau, seulement comment ces données doivent se présenter.

WSDL : (*Web Services Description Language*) Langage de description des services Web, c'est un protocole reposant sur XML qui procure une description complète d'un service Web (sa localisation, une liste des fonctions publiées, des arguments qu'elles attendent et des valeurs qu'elles retournent). La disponibilité de ce fichier suffit à un client SOAP pour lui permettre d'interroger n'importe quel service Web publié, sans aucune connaissance préalable de ce service.

Excel Add-In : un Add-In (généralement écrit en VBA et prenant comme extension .xla), est une extension apportant à Excel de nouvelles fonctionnalités. Il n'est pas lié à une feuille de calcul en particulier.

V - Déploiement du connecteur

a) Côté serveur

La base de données 4D fournie tourne quelque part sur le réseau et son serveur Web a été activé. Voilà le seul pré-requis pour tester la démonstration.

Pour vérifier que tout fonctionne correctement, essayez d'accéder au fichier WSDL généré automatiquement par 4D. Il devrait se trouver à l'adresse suivante : "http://votreIP:8080/4dwsdl"

> *Note : dans cet exemple, le serveur Web est publié sur le port 8080 pour éviter d'avoir à se connecter en "root" sous OSX. Cela ne change rien au niveau des fonctionnalités, il suffit de se rappeler d'ajouter ":8080" aux URLs saisies.*

b) Côté client

D'abord, assurez-vous d'avoir correctement installé le Microsoft SOAP Toolkit (3.0 minimum). Ensuite, vous devrez installer l'Add-In Excel fourni. Cela s'effectue simplement en glissant-déposant le fichier .xla dans le bon dossier. Il y a deux endroits appropriés pour cela :

Niveau global : par exemple dans "C:\Program Files\OfficeXP\Office10\XLStart" pour une installation normale sous XP.

Par utilisateur : par exemple dans "C:\Documents and Settings\bibi\Application Data\Microsoft\AddIns" pour l'utilisateur nommé bibi.

Enfin, lancez Excel et ouvrez une nouvelle feuille de calcul.

Vous devriez maintenant voir un nouveau menu nommé "SOAP Connectivité" dans Excel, juste avant le menu d'aide. Sélectionnez l'item "Préférences". Cela devrait ouvrir un dialogue, où vous saisirez l'URL d'accès au WSDL de votre Web Service : quelque chose dans le genre de "http://votreIP:8080/4dwsdl".

Si vous désirez étudier le code du connecteur, allez dans "Outils/Macro/Visual Basic Editor".

> *Note : Si Excel vous prévient qu'il ne peut pas exécuter de macro à cause du niveau de sécurité, il se peut que vous deviez changer les réglages dans "Outils/Macro/Sécurité#".*

VI - Implémentation côté serveur dans 4D

Pour le besoin de cette note technique, j'ai choisi de publier comme service Web un jeu de données extrêmement limité.

L'intérêt principal réside dans la communication qui intervient entre Excel et la base de données, non dans la base de données elle-même.

Ainsi nous disposons simplement d'une table, de quatre champs et de quelques deux milles entrées générées automatiquement :

[TableOne]

· Nom (texte)

· Score (réelle)

· DoB (date)

· Categorie (entier long)

VII - Description de SOAP_GetDataForCategory

Cette méthode, la seule publiée dans cet exemple, accepte un paramètre (entier long) et retourne quatre types de tableaux différents (texte, réel, date et entier long).

Lors de la réception d'une requête, notre code prend en compte le paramètre reçu, effectue une recherche sur toutes les entrées correspondant à la catégorie demandée, et renvoie les données en utilisant des tableaux et la commande **SELECTION VERS TABLEAU**.

Pour un développeur 4D le code de gestion des données est limpide. Nous concentrerons donc nos explications sur le code mettant en #uvre SOAP.

Voici une explication ligne à ligne :

Comme indiqué précédemment, la méthode accepte seulement une valeur de type entier long.

```
C_ENTIER LONG($1)

DECLARATION SOAP($1;Est un entier long;SOAP Entrée;"category")
```

L'appel à **DECLARATION SOAP** n'est pas indispensable. Pour des types de base (c-a-d différents de tableaux), les directives de compilation sont suffisantes pour publier la méthode. Cependant, le recours systématique à **DECLARATION SOAP** pour tous les arguments constitue une bonne pratique (pour des raisons d'homogénéité). En outre, cela vous offre la possibilité de personnaliser le nom sous lequel cet argument sera publié ("category" au lieu du nom attribué automatiquement par 4D "FourDArg1").

Ensuite, effectuons un minimum de contrôle sur la valeur de $1, afin de s'assurer qu'il s'agit d'un ID valide (la valeur maximale étant de 28).

```
Si ($1>28)

   ENVOYER ERREUR SOAP (SOAP erreur client ;"La catégorie doit être inférieure à 28")

Fin de si
```

En cas de problème, nous renvoyons une erreur SOAP. Cette fonctionnalité intéressante du protocole nous permet de renvoyer des erreurs sans retourner de résultat. A l'aide des constantes fournies vous pouvez également désigner un coupable (le client ou le serveur) dans le premier argument passé à **ENVOYER ERREUR SOAP**. Le second paramètre décrira l'erreur par une chaîne de caractères.

Les lignes suivantes sont évidentes :

```
TOUT SELECTIONNER([TableOne])

Si ($1#0)

   CHERCHER([TableOne];[TableOne]Categorie=$1)

Fin de si
```

```
$len:=Enregistrements trouves([TableOne])
```

Rendu à ce point, nous connaissons la taille des tableaux que nous allons retourner. Nous pouvons donc les déclarer par les directives de compilation et les publier en faisant appel à **DECLARATION SOAP**.

```
TABLEAU TEXTE(retArray1;$len)

DECLARATION SOAP(retArray1;Est un tableau texte ;SOAP sortie ;"arrayName")

TABLEAU REEL(retArray2;$len)

DECLARATION SOAP(retArray2;Est un tableau numérique ;SOAP sortie ;"arrayReal")

TABLEAU DATE(retArray3;$len)

DECLARATION SOAP(retArray3;Est un tableau date ;SOAP sortie ;"arrayDate")

TABLEAU ENTIER LONG(retArray4;$len)

DECLARATION SOAP(retArray4;Est un tableau entierlong ;SOAP sortie ;"arrayLongint")
```

Cette fois, l'utilisation de **DECLARATION SOAP** est obligatoire pour deux raisons:

```
# Nous manipulons des tableaux
```

ET

```
# Nous retournons plusieurs valeurs.
```

Les dernières lignes ne sont pas beaucoup plus complexes : elles remplissent les tableaux et génèrent une erreur si aucun enregistrement n'a été trouvé.

```
Si ($len#0)
   SELECTION VERS TABLEAU([TableOne]Nom;retArray1;[TableOne]Score;retArray2;
   [TableOne]DoB;retArray3;[TableOne]Categorie;retArray4)
Sinon
   ENVOYER ERREUR SOAP(SOAP erreur serveur ;"Aucun enregistrement trouvé pour cette catégorie")
Fin de si
```

a) Activation de la publication du service Web

Cette partie est particulièrement enfantine. Il suffit de réaliser les trois tâches suivantes :

Activer la publication de service Web (dans les préférences de votre base de données, section Web, partie Web Services).

Publier les méthodes composant le service Web. Un passage par les propriétés de la méthode ciblée suffit. L'icône des méthodes publiées diffère dans la fenêtre de l'Explorateur.

Démarrez le serveur Web.

Votre service Web est maintenant officiellement ouvert !

b) Personnalisation du nom du service Web et de son espace de nom

Si vous le souhaitez, vous pouvez personnaliser quelques éléments publiés automatiquement dans le WSDL que 4D génère. La section précédente vous a montré comment personnaliser le nom de les paramètres d'entrée et de sortie, en utilisant **DECLARATION SOAP**.

Vous pouvez aussi paramétrer le nom du service publié et l'espace de nom utilisé par le message SOAP. Ces paramètre sont accessibles dans les préférences de la base de données en sélectionnant la ligne Web Services de la section Web.

Vos pouvez également ajouter de la documentation à vos méthodes en cliquant sur le bouton radio "Commentaires" de l'Explorateur. Ces commentaires seront publiés sur le Web dans le WSDL.

> *Note : Pour les services Web, les noms des méthodes et des arguments sont soumis à des restrictions par le protocole SOAP. Reportez-vous à la documentation 4D concernant les services Web pour des compléments à ce sujet. Mon conseil consiste à se limiter aux caractères de l'us-ascii en n'employant ni l'espace, ni les accents, et aucun chiffre comme premier caractère.*

VIII - Implémentation côté client en VBA

Un Add-In Excel consiste généralement en du code VBA enregistré avec l'extension .xla. Un utilisateur Excel peut ensuite choisir de charger ou non telle ou telle extension.

L'Add-In fourni se composa de quatre parties :

Un dialogue très simple, nommé UserForm, qui permet d'enregistrer l'URL du WSDL du 4D Server cible.

Un module nommé Init, qui contient du code pour initialiser et dé-initialiser les menus ajoutés à Excel.

Un module nommé JustDoIt, qui contient le code réel d'appel au serveur 4D et qui remplit la feuille de données avec les données reçues.

Un module de classe nommé clsws_ExampleOfWebService, proxy généré automatiquement par le *SOAP* Toolkit après analyse du WSDL de votre service Web.

Le dialogue de préférences est un formulaire classique VB. La seule saisie consiste en l'URL du WSDL à interroger (généré par 4D), cette valeur est enregistrée dans la variable globale gServerURL.

Le module Init contient du code intéressant pour l'initialisation de l'Add-In Excel, en particulier concernant l'ajout de menus à Excel. Les méthodes nommés Auto_Open et Auto_Close sont automatiquement appelées quand l'Add-In est chargé ou libéré.

Les points intéressants du code VB sont principalement situés dans le module JustDoIt, plus précisément dans la méthode RunMenu1. En particulier vous y trouverez comment appeler la méthode cible (à l'aide de l'objet MyCall), comment retourner des tableaux (en utilisant le type Variant) et comment remplir les cellules de la feuille de calcul par les valeurs retournées (en utilisant la propriété ActiveCell et CurrentSheet.Cells).

Le quatrième module a été particulièrement facile à créer grâce au toolkit. Dans Visual Basic Editor, sélectionnez "Outils/Web Service Reference". Cliquez sur le bouton radio "URL du Service Web" pour entrer manuellement l'URL du fichier WSDL généré par 4D. Analysez le, sélectionnez le Web Service reconnu et cliquez sur le bouton Ajouter. La classe doit alors être générée.

Vous pouvez utiliser ce module de classe comme si le serveur ne changeait pas de localisation entre le moment où vous générez la classe et celui où vous exécutez votre code. Mais cela n'est pas exactement notre cas, nous désirons pouvoir paramétrer la localisation du serveur à interroger. C'est pourquoi nous avons effectuer un seul changement dans le code généré (chercher la variable nommée gServerURL pour le trouver).

IX - Considérations avancées

VB se comporte d'une manière inhabituelle pour un client SOAP : à chaque fois que la classe interne SOAP est initialisée, le WSDL est ré-interrogé et ré-analysé. Cela peut être considéré comme une fonctionnalité ou un sérieux problème. L'idée sous-jacente est, bien sûr, de s'assurer que le service Web que nous sommes sur le point d'appeler se présente de la même façon que lors de son analyse.

Mais il y a des contreparties en terme de performance car cela nécessite beaucoup de CPU et de trafic réseau pour peu de résultat; de plus sans réel intérêt car si vous changez votre service du côté serveur, VB se plaindra quelquefois sans que l'appel RPC ait été lui-même affecté. Mon avis sur ce sujet est que cette "fonctionnalité" devrait être désactivée par défaut. Un moyen de contourner ce problème consiste à enregistrer en local le fichier WSDL et à l'analyser depuis le disque et non depuis le réseau.

Il y a une autre caractéristique cachée dans l'implémentation VBA qui peut se révéler un avantage : vous avez déjà noté que lorsque nous manipulons des tableaux, le code VB manipule quant à lui des variables de type Variant. Nous ne spécifions par à l'avance le type que nous attendons réellement en retour. En raison du point exposé précédemment (la ré-analyse systématique du WSDL), on pourrait penser que si le type de données retourné en pratique diffère de celui décrit dans le WSDL, VB ne serait pas content. Hé bien, en fait on pourrait se tromper ! D'après mes tests, cela semble marcher. Cela signifie que nous pourrions écrire une SEULE méthode 4D qui retourne différents types de tableaux suivant la requête soumise, ce qui est assez séduisant.

Cependant, j'ai décidé de ne pas tirer parti de cette "caractéristique", car elle ressemble beaucoup à un vilain "hack" dont nous ne pouvons pas être certain du support par la prochaine version du toolkit. Mais pour ceux d'entre vous qui aimez vivre dangereusement, sentez-vous libre d'essayer cette astuce!

X - Considérations sur les performances

Ce genre de solutions est idéal pour des jeux de données de petite à moyenne taille (typiquement plusieurs milliers de lignes). La montée en charge sera correcte, même avec plusieurs clients Excel se connectant simultanément. Cependant, si vous désirez échanger de grandes quantités de données, vous pouvez être amené à étudier d'autres solutions, car SOAP est plutôt gourmand en ressources mémoire et réseau.

XI - Aller plus loin avec le Connecteur#

a) le rendre plus générique

Actuellement, le connecteur "connaît" la base de données à laquelle il s'adresse (son nom, les noms des champs, etc.). Une amélioration significative consisterait à ce que le Connecteur découvre la base lors de l'exécution et laisse l'utilisateur choisir quelle table il veut gérer. Une manière de faire consisterait à fournir deux fonctions procurant de l'information sur la base de données.

Vous pouvez également modifier le code afin qu'il mette à jour des données 4D depuis Excel. Pour cela, il suffit d'inverser la logique mise en #uvre : créez et publiez une nouvelle méthode 4D qui accepte des tableaux en entrée. Générez un nouveau proxy VB, et appelez le service Web en lui passant des données provenant de la feuille de calcul Excel.

b) Considérations de sécurité

Deux besoins sont recensés ici, vous pouvez souhaiter :

encrypter l'échange de données.

avoir plus de contrôle sur qui accède à quoi.

La premier point pourrait être facilement résolu en utilisant une connexion SSL qui encrypterait tout le trafic réseau.

Pour le second point, un mécanisme d'authentification serait nécessaire. Au moment de cette rédaction, je n'ai pas trouvé de moyen d'utiliser le mécanisme d'authentification basique d'HTTP dans VB.

Note du rédacteur

La note "Building Secure Web Services with Microsoft SOAP Toolkit 2.0" apporte des informations sur ce sujet.

http://msdn.microsoft.com/library/en-us/dnsoap/html/soapsecurity.asp?frame=true

Nous pourrions construire une solution sur mesure, en utilisant deux nouvelles fonctions publiées par 4D telles que login et logout. Un identificateur de session retourné par l'appel à la fonction login pourrait être utilisé pour chaque transaction et expirerait au bout d'un délai paramétrable.

c) Add-In pour Word

Comme Excel, Word supporte l'addition d'Add-Ins sur mesure. Même si l'utilisation d'un tel type de connecteur est moins évidente qu'avec Excel, il est assez facile d'imaginer des situations dans lesquelles vous souhaiteriez appeler un service Web et recevoir directement en retour les données dans Word (import automatique de table, images, téléchargement d'images depuis une base de données). En particulier, une fusion afin de réaliser un mailing à partir de données provenant de 4D présenterait de l'intérêt.

XII - Conclusion

4ème Dimension 2003 rend extrêmement aisé l'exposition de certaines de vos méthodes en tant que service Web. Cela ouvre la porte à beaucoup de nouvelles possibilités pour les applications *end-user*, comme Excel, afin d'échanger des informations avec vos bases de données 4D.

XIII - Base exemple

Téléchargez la base exemple :

Base pour Windows

Base pour Mac OS