Web Services avec paramètre d'entrée complexe (Part. 1)

Par
Dave DELL'AQUILA, Senior evangelist, 4D Inc.
Note technique 4D-200311-33-FR
Version 1

Date 1 Novembre 2003

Résumé

Cette note technique est la première partie d'une note en deux parties expliquant comment générer un paramètre d'entrée SOAP correcte utilisable avec les Web Services qui attendent un type d'entrée complexe.

4D Notes techniques

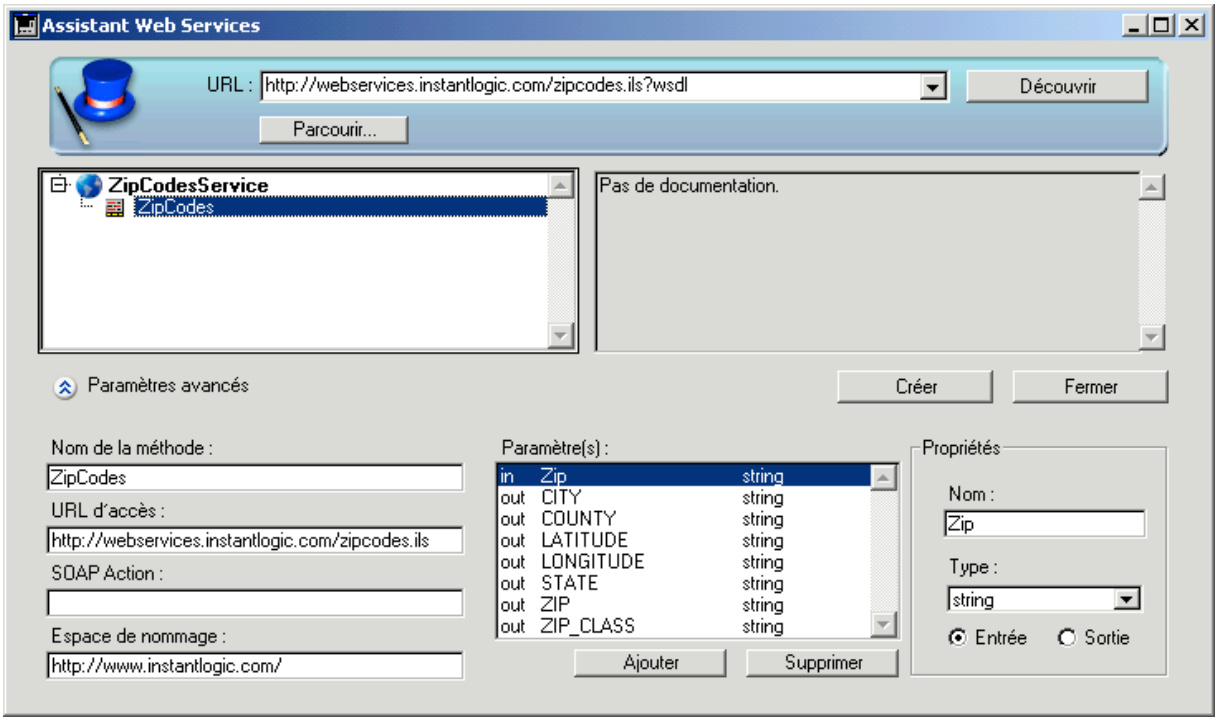
Web Services avec paramètre d'entrée complexe (Part. 1)

Cette note technique est la première partie d'une note en deux parties expliquant comment générer un paramètre d'entrée SOAP correcte utilisable avec les Web Services qui attendent un type d'entrée complexe.

Introduction

Les Web Services sont des fonctions indépendantes de plate-forme qui peuvent être incorporés dans d'autres applications à travers des protocoles standards. Les Web Services permettent d'offrir des services à n'importe quelle application souhaitant les utiliser. Des conventions standards pour les paramètres attendus et retournés par le Web Service ont été développés et adoptés par les acteurs majeurs du marché.

Pour que cela fonctionne, tout ce qui est nécessaire pour invoquer le Web Service (par n'importe quel client Web Service) est contenu dans un document XML appelé "WSDL" (*Web Services Description Language*). Le wizard Web Service de 4D analyse le fichier WSDL pour n'importe quel Web Service publié, et affiche les informations spécifiques à l'interface SOAP du Web Service de façon très lisible :



Dans l'exemple ci-dessus, notez la liste des paramètres d'entrée et de sortie. Le Web Service décrit par le WSDL indique qu'il utilise des paramètres de style RPC (*Remote Procedure Calls*). Chaque paramètre est listé avec sa direction (entrée ou sortie - *in* ou *out*), son nom et son type SOAP. Le wizard utilise cette information (obtenue grâce au WSDL) pour déterminer comment construire la méthode proxy exactement comme n'importe quelle méthode 4D, passer les valeurs à celle-ci et (optionnellement) recevoir les valeurs retournées de celle-ci.

A part cliquer sur le bouton **Créer** pour générer la méthode proxy, il n'y a rien de plus à faire pour envoyer des valeurs et en recevoir en retour du Web Service. Le wizard génère automatiquement la méthode 4D pour communiquer avec le Web Service :

```
`
` ZipCodes
` url:  http://webservices.instantlogic.com/zipcodes.ils?wsdl
` Méthode générée automatiquement par l'assistant Web Services de 4D.
`

` ----------------------------------------------------------------
C_TEXTE($1)
C_TEXTE(ret1)
C_TEXTE(ret2)
C_TEXTE(ret3)
C_TEXTE(ret4)
C_TEXTE(ret5)
C_TEXTE(ret6)
C_TEXTE(ret7)
C_TEXTE(ret8)

FIXER PARAMETRE WEB SERVICE("Zip";$1)

APPELER WEB SERVICE("http://webservices.instantlogic.com/zipcodes.ils";"";
        "ZipCodes";"http://www.instantlogic.com/";Web Service dynamique )

Si (OK=1)
LIRE RESULTAT WEB SERVICE(ret1;"CITY")
LIRE RESULTAT WEB SERVICE(ret2;"COUNTY")
LIRE RESULTAT WEB SERVICE(ret3;"LATITUDE")
LIRE RESULTAT WEB SERVICE(ret4;"LONGITUDE")
LIRE RESULTAT WEB SERVICE(ret5;"STATE")
LIRE RESULTAT WEB SERVICE(ret6;"ZIP")
LIRE RESULTAT WEB SERVICE(ret7;"ZIP_CLASS")
LIRE RESULTAT WEB SERVICE(ret8;"Error";*)   ` Libération de la mémoire après retour de la valeur.

Fin de si
```

Tous les paramètres à envoyer au Web Service transitent par la méthode proxy comme des paramètres standards 4D. La commande **FIXER PARAMETRE WEB SERVICE** "parque" chaque paramètre devant être passé au Web Service par l'intermédiaire de la commande **APPELER WEB SERVICE**. Une fois que tous les paramètres sont mis en place, **APPELER WEB SERVICE** prépare alors la requête correctement formatée pour le Web Service à lui envoyer. La requête est un document XML qui ressemble à ceci :

```
<?xml version="1.0" encoding="UTF-8" standalone="no"?>
<SOAP-ENV:Envelope xmlns:SOAP-ENV="
http://schemas.xmlsoap.org/soap/envelope/"
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:s0="http://www.instantlogic.com/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" >
<SOAP-ENV:Body>
<mns:ZipCodes xmlns:mns="http://www.instantlogic.com/" SOAP-ENV:
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
<Zip xsi:type="xsd:string">95128</Zip></mns:ZipCodes>
</SOAP-ENV:Body>
</SOAP-ENV:Envelope>
```

La requête ci-dessus est envoyée au Web Service pour être exécutée et retourner une réponse, également au format XML :

```
<env:Envelope xmlns:env="http://schemas.xmlsoap.org/soap/envelope/"
```

```
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:s0="http://www.instantlogic.com/"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/">
<env:Body
env:encodingStyle="http://schemas.xmlsoap.org/soap/encoding/">
<s0:ZipCodesResponse>
<CITY xsi:type="xsd:string">SAN JOSE</CITY>
<COUNTY xsi:type="xsd:string">SANTA CLARA</COUNTY>
<LATITUDE xsi:type="xsd:string">+37.189396</LATITUDE>
<LONGITUDE xsi:type="xsd:string">-121.705327</LONGITUDE>
<STATE xsi:type="xsd:string">CA</STATE>
<ZIP xsi:type="xsd:string">95128</ZIP>
<ZIP_CLASS xsi:type="xsd:string">STANDARD</ZIP_CLASS>
<Error xsi:type="xsd:string"></Error>
</s0:ZipCodesResponse>
</env:Body>
</env:Envelope>
```

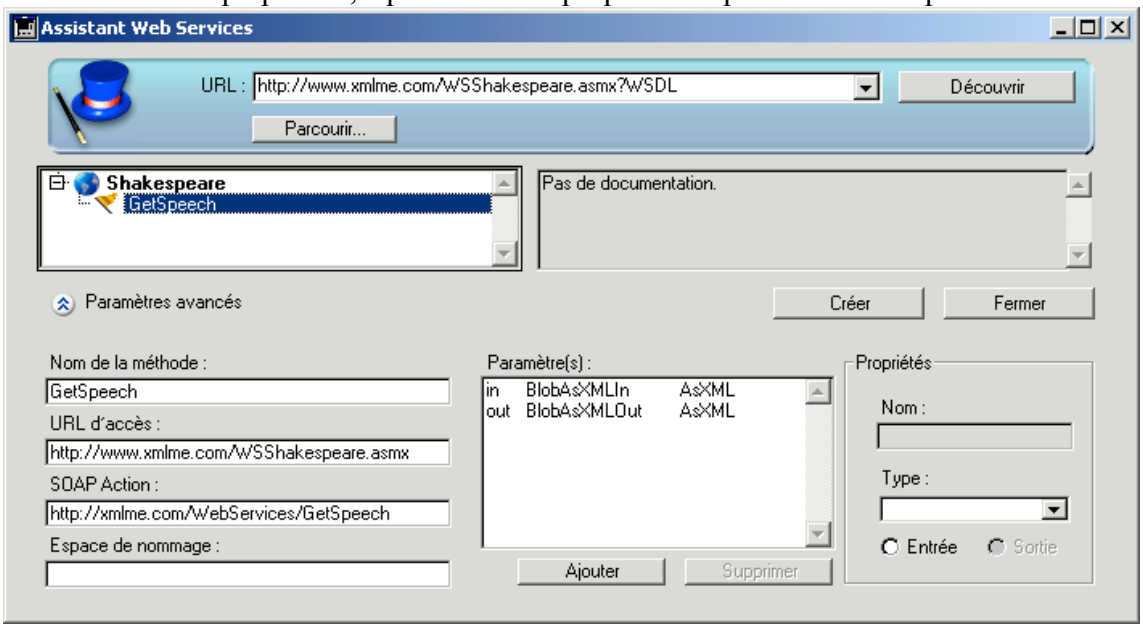
La requête et la réponse sont alternativement envoyées au et du Web Service dans une enveloppe SOAP, très semblable à un "paquet" qui serait délivré par un service postal. Notez que dans l'exemple ci-dessus, les paramètres séparés sont tous retournés dans la partie **Body** comme paramètres individuels.

D'autres Web Services, qui utilisent des paramètres de styles DOC fonctionnent d'une manière différente, demandant un peu plus de travail de la part du développpeur.

Les Web Services qui attendent un style DOC, ou des paramètres "complexes", attendent un unique paramètre d'entrée, utilisant XML, et peuvent retourner un unique paramètre de sortie, utilisant aussi XML. Ceci permet à un jeu plus complexe de données d'être reçues ou renvoyées d'un Web Service qui utilise cette méthode de passage de données entre l'application Web Service serveur et l'application cliente.

Le wizard 4D Web Services détecte et affiche les paramètres de style DOC en affichant un petit drapeau à côté du nom de la méthode.

Voici ci-dessous un exemple d'analyse du WSDL
http://www.xmlme.com/WSShakespeare.asmx?WSDL
un Web Service qui permet, à partir d'une réplique d'une pièce de Shakespeare de connaître de nom de la pièce :



Notez que dans l'écran ci-dessus, il y a seulement deux paramètres listés : un paramètre d'entrée appelé *BlobAsXMLIn* et un paramètre de sortie appelé *BlobAsXMLOut*.

Lorsque vous utlisez les Web Services qui attendent un paramètre d'entrée de type DOC, c'est à vous de vérifier que la requête SOAP est correcte. Plus précisément, les informations que vous envoyez à la commande **FIXER PARAMETRE WEB SERVICE** sont seulement le contenu de l'élément **body** de la requête. 4D créera tous les autres éléments nécessaires à la requête, incluant le **header** XML, le **SOAP-ENV :Envelope element** et les définitions **namespace (espace de noms)**, **SOAP-ENV:Body element**, et les éléments correspondants de fermeture.

L'exemple suivant monte une requête de type DOC :

```
<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
- <SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:http="http://schemas.xmlsoap.org/wsdl/http/"
    xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
    xmlns:s="http://www.w3.org/2001/XMLSchema"
    xmlns:s0="http://xmlme.com/WebServices"
    xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
    xmlns:tm="http://microsoft.com/wsdl/mime/textMatching/"
    xmlns:mime="http://schemas.xmlsoap.org/wsdl/mime/"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  - <SOAP-ENV:Body>
    - <s0:GetSpeech xmlns:s0="http://xmlme.com/WebServices">
        <s0:Request>my kingdom for a horse</s0:Request>
      </s0:GetSpeech>
    </SOAP-ENV:Body>
  </SOAP-ENV:Envelope>
```

Notez que le contenu de la requête est vraiment similaire à une requête de type RPC vue auparavant. Seulement les **espaces de noms (namespaces)** et le conteu du **SOAP body** sont différents.
Pour envoyer les données au Web Service, vous devez simplement fournir la partie **SOAP-ENV:Body element** :

```
- <s0:GetSpeech xmlns:s0="http://xmlme.com/WebServices">
    <s0:Request>my kingdom for a horse</s0:Request>
  </s0:GetSpeech>
```

Si vous utilisez l'assistant Web Services de 4ème Dimension pour créer votre méthode proxy pour le Web Service, vous obtiendrez la méthode suivante :

```
` proxy_GetSpeech

` url: http://www.xmlme.com/WSShakespeare.asmx?WSDL
` Méthode générée automatiquement par l'assistant Web Services de 4D.
`

` ------------------------------------------------------------
C_BLOB($1)
C_BLOB($0)

FIXER PARAMETRE WEB SERVICE("BlobAsXMLIn";$1)
```

```
APPELER WEB SERVICE("http://www.xmlme.com/WSShakespeare.asmx";"http://xmlme.com/
    WebServices/GetSpeech";"GetSpeech";"";Web Service manuel )

Si (OK=1)
LIRE RESULTAT WEB SERVICE($0;"BlobAsXMLOut";*)  ` Libération de la mémoire après retour de la valeur.

Fin de si
```

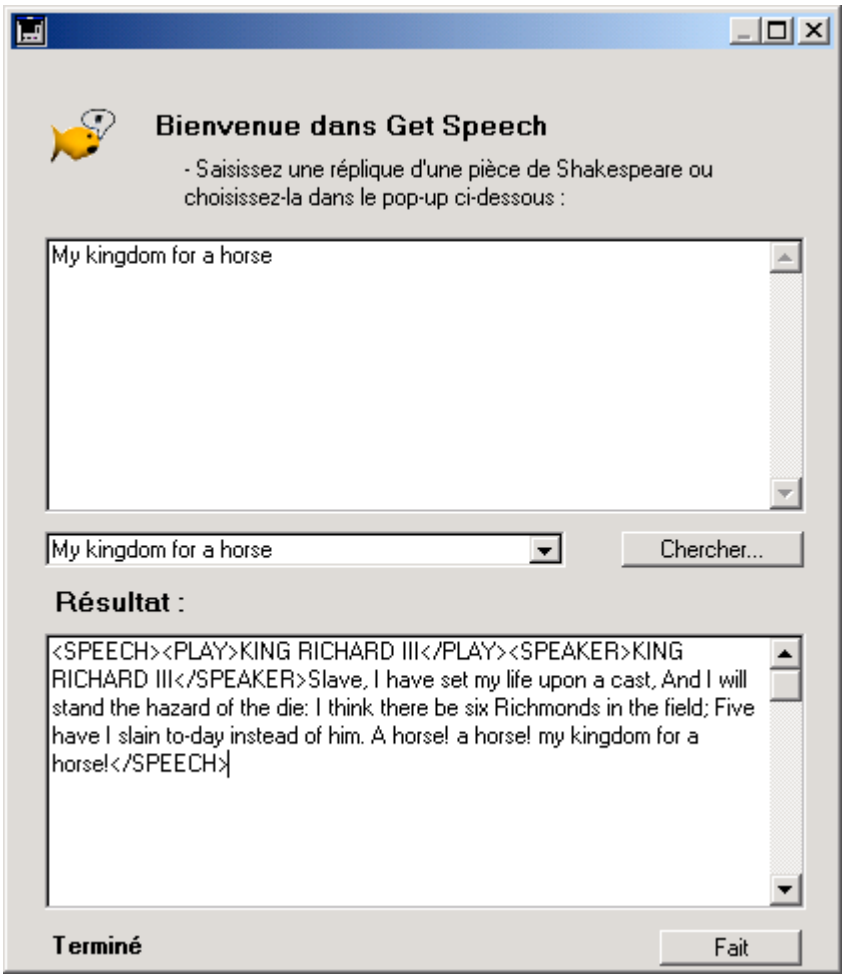
Pour appeler le Web Service de votre application, vous pouvez utiliser le code 4D ci-dessous :

```
` Méthode projet : GP_callSOAP

` Description: Appelle la méthode proxy GetSpeech et transmet la requête.

C_BLOB(<>returnBlob;$tempBlob)
C_ENTIER LONG($2;$TargetProcessID)
C_TEXTE($1;<>vtResult;$tempText;$returnText;$tempRequest)

$tempRequest:=$1
$TargetProcessID:=$2
$tempText:="<s0:GetSpeech  xmlns:s0=\"http://xmlme.com/WebServices\">"+"<s0:Request>"
    +$tempRequest+"</s0:Request></s0:GetSpeech>"
TEXTE VERS BLOB($tempText;$tempblob;Texte sans longueur )
<>returnBlob:=proxy_GetSpeech ($tempBlob;"")
<>vtResult:=BLOB vers texte(<>returnBlob;Texte sans longueur )

APPELER PROCESS($TargetProcessID)
```

Le code ci-dessus assigne simplement une chaîne à la variable $temText, convertit le texte en blob et envoie celui-ci à la méthode proxy. Le quatrième paramètre, Web Service manuel , de la commande **APPELER WEB SERVICE**, insère le contenu du blob dans la partie **SOAP-ENV:Body** de la requête avant de l'envoyer au Web Service. Le résultat, aussi un blob, est retourné par la méthode proxy. Le blob contient juste la partie **body** de la réponse SOAP, et vous devez manuellement analyser son contenu.

La base de démonstration :

Choisissez une réplique dans le pop-up situé au milieu du dialogue, et cliquez sur Chercher.



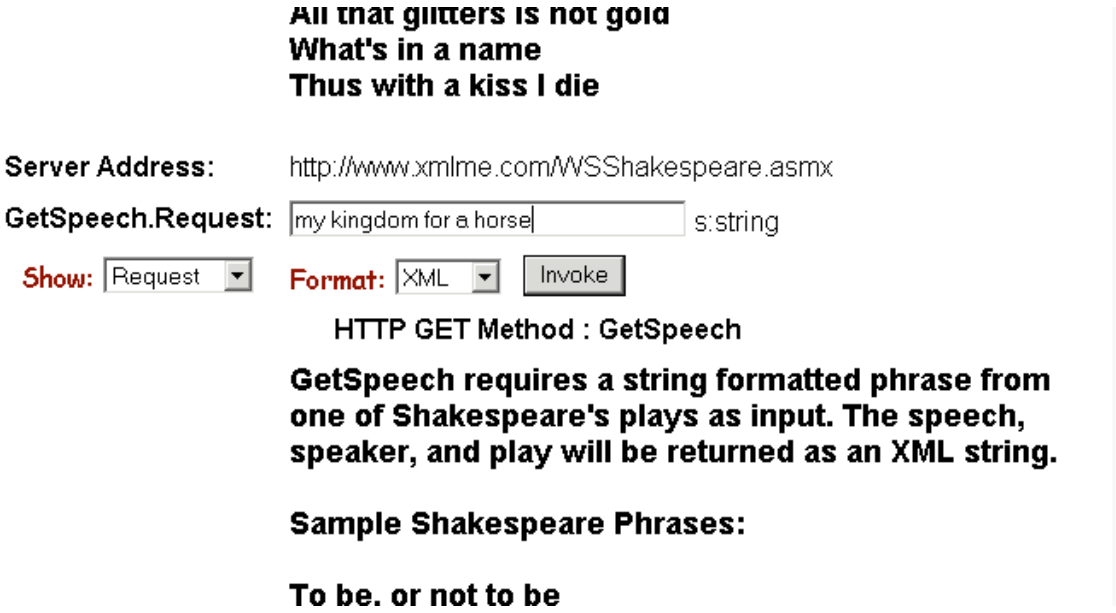
Vous obtenez une chaîne que vous pouvez alors analyser pour récupérer les informations souhaitées.

Déterminer comment formater des paramètres d'entrée complexes

Il y a différentes façons de déterminer exactement ce que vous devez passer à une méthode proxy attendant un paramètre complexe. Une façon simple est d'utiliser un client SOAP générique qui permet de voir la requêtre proprement formée avant de l'envoyer au Web Service. Par exemple :

http://www.soapclient.com/soaptest.html

Le WSDL analysé, vous pouvez lancer une requête et choisir de visualiser soit la requête, soit la réponse :



Vous pouvez voir la requête avant de l'envoyer au Web Service, ce qui vous permet de voir le format du **SOAP-EN:Body element**. Des navigateurs récents vous montrent seulement les valeurs significatives de la requête. Vous pouvez récupérer le code source de la page et le visualiser dans Internet Explorer ou tout autre logiciel capable de lire du XML. Ci-dessous la requête complète telle qu'envoyée au Web Service :

```
<?xml version="1.0" encoding="UTF-8" standalone="no" ?>
- <SOAP-ENV:Envelope xmlns:SOAP-ENV="http://schemas.xmlsoap.org/soap/envelope/"
    xmlns:http="http://schemas.xmlsoap.org/wsdl/http/"
    xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
    xmlns:s="http://www.w3.org/2001/XMLSchema"
    xmlns:s0="http://xmlme.com/WebServices"
    xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
    xmlns:tm="http://microsoft.com/wsdl/mime/textMatching/"
    xmlns:mime="http://schemas.xmlsoap.org/wsdl/mime/"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema">
  - <SOAP-ENV:Body>
    - <s0:GetSpeech xmlns:s0="http://xmlme.com/WebServices">
        <s0:Request>my kingdom for a horse</s0:Request>
      </s0:GetSpeech>
    </SOAP-ENV:Body>
  </SOAP-ENV:Envelope>
```

Le **SOAP-ENV:body element** inclut typiquement les éléments suivants :

<Préfixe : Nom de la Méthode Espace de Nom>	Elément d'ouverture
<Préfixe : Nom du Paramètre> Valeur </Préfixe : Nom du Paramètre>	Répéter pour chaque paramètre
<Préfixe : Nom de la Méthode>	Elément de fermeture

Une autre façon pour déterminer comment formater des paramètres d'entrée complexes est d'inspecter manuellement le WSDL (ou de créer une méthode qui le fasse). Dans la partie 2 de cette note technique, nous examinerons comment déterminer ces paramètres attendus par un Web Services de type DOC.

Conclusion :

Beaucoup de Web Services actuels sont conçus en utilisant les paramètres d'entrée complexes. La première version de l'implémentation des Web Services de 4D rend extrémement aisée d'utilisation des Web Services de style RPC sans programmation particulière de la part du développeur. Les Web Services de style DOC requièrent de la part du développeur la construction d'un blob qui contient l'élément **body** avec la description XML des paramètres attendus par le Web Service. Dans la seconde partie de cette note, nous examinerons comment extraire les informations du fichier WSDL du Service Web.