The REBOL Documentation Project

-- FR - Documentation REBOL - Tutoriels --

Tutoriels

REBOL/Services pour les nuls (1) - La définition

REBOLtof

Première publication : 4 janvier 2006, et mis en ligne le mercredi 4 janvier 2006

Résumé :

Premier d'une série visant à documenter les REBOL/Services, cet article va se concentrer sur leur définition et sur leurs usages.

Premier d'une série visant à documenter les REBOL/Services, cet article va se concentrer sur leur définition et sur leurs usages.

Version : 1.0
Date : 19-dec-2005

1 Historique

Date	Version	Commentaires	Auteur	Email
19-dec-2005	1.0	Version intiale	REBOLtof	reboltof—at—yahoo//dot//com

2 Introduction

Bienvenue dans le monde sauvage, fascinant et merveilleux des applications communicantes !

2.1 A propos de cette série d'articles

Cette série d'articles va tenter de vous initier à l'utilisation des REBOL/Services, dernière innovation de REBOL Technologies pour le langage de programmation REBOL.

2.2 Comment lire ces articles ?

Cette série d'articles abordera le sujet de manière progressive, en soulignant et définissant les concepts de bases nécessaires à la compréhension des REBOL/Services.

La série se terminera par des articles essentiellement techniques. Il y aura de la sueur et du cambouis :-) Un bref coup d'oeil sur un article précédent vous permettra sans doute d'éclaircir certains points restés obscurs.

2.3 Pré requis pour comprendre les articles

- Vous êtes capable d'allumer votre ordinateur.
- Vous avez des connaissances (même basiques) du langage de programmation REBOL.
- Vous souhaitez comprendre ce que les REBOL/Services représentent pour vous.

IMPORTANT

Les REBOL/Services en sont maintenant au stade de développement ALPHA. Cela signifie que nous pouvons les étudier, les utiliser, tout en sachant que certains bugs peuvent encore être présents et que certaines fonctionnalités peuvent encore être appelées à évoluer.

A ce stade-ci, nous sommes encore obligés de nous procurer les scripts nécessaires à l'utilisation des /Services, et de les inclure dans nos scripts. Dans un futur que nous espérons proche, les /Services seront inclus dans le code de l'interpréteur.

3 REBOL/Services, qu'est-ce que c'est ?

3.1 Le concept de "fonction"

De nos jour, la grosse majorité des programmeurs sont familiers avec la notion de "fonction" : il s'agit d'un regroupement logique de lignes de code qui fournissent une ou plusieurs "fonctionnalités". Afin de fournir celles-ci, la fonction a parfois besoin de données à traiter (les paramètres), et parfois pas.

Pour illustrer l'idée, prenons simplement une fonction qui met à disposition la fonctionnalité de *compter le nombre de lettres du mot passé en paramètre* :

```
>> compte-lettre: func [mot][return length? form mot]
>> compte-lettre "hello"
== 5
```

Nous y retrouvons

- *le nom de la fonction* : "compte-lettre"
- *le paramètre de la fonction* : "mot"
- *le corps de la fonction*, ou comment la fonctionnalité est implémentée : "return length ? form mot"

3.2 Des fonctions gourmandes en ressources système

Imaginons que nous devions utiliser une fonction devant appliquer certaines transformation sur des images haute résolution, comme par exemple l'effacement de "parasites". Ce type de fonctionnalité exige traditionnellement beaucoup de la puissance de calcul du processeur. Utiliser cette fonction sur notre PC risque de bloquer son fonctionnement un temps, rendant impossible d'autres utilisations.

Une des solutions possibles à ce problème est d'exécuter cette fonction sur un PC plus puissant, et de récupérer les résultats. Nous pouvons évidement mettre l'image à traiter sur un CD-ROM ou une clef USB, nous déplacer jusqu'à ce PC, y charger l'image, lancer la fonction, et, quand le traitement est terminé, effectuer l'opération inverse pour récupérer les données, mais il y a plus simple !

3.3 Utilisation de fonctionnalité distante

Imaginons maintenant que nous pouvions disposer d'un mécanisme nous permettant *de notre PC*, d'exécuter une fonction *sur un PC distant...* Notre problème est alors résolu : nous pouvons, au moment où nous en avons besoin, faire appel à la fonction distante, attendre le résultat (qui vient rapidement, le PC distant étant plus puissant que le nôtre), et continuer notre travail...

3.4 Fonctionnalité distante et ... asynchrone

Fantasmons encore plus : imaginons maintenant que nous n'ayant pas besoin du résultat de la fonctionnalité distante directement après l'avoir sollicitée, mais quelque temps après. Ce qui nous permet, durant le temps d'attente, d'effectuer d'autres travaux... Il nous suffirait de demander à cette fonction distante de nous envoyer un signal afin de nous avertir de la fin du travail !

3.5 Abracadabra, REBOL/Services est là !

Vous le sentiez venir, fins psychologues que vous êtes !

REBOL/Services nous offre la possibilité de créer ce type de fonctions distantes, de les appeler, de recueillir des résultats, d'attendre la fin du traitement ou pas ...

3.6 Comment fonctionne REBOL/Services ?

L'architecture interne des /Services fera l'objet d'autres articles. A ce stade-ci nous pouvons déjà mentionner que le modèle utilisé est un modèle Client/Serveur des plus classiques.

Le **Serveur** fournira les différentes fonctionnalités créées. Il s'agit d'un script qui est lancé sur le PC distant et qui continue de tourner, constamment à l'écoute d'un port de communication (pour rappel, les "ports" sont une sorte de "tampon" ou de zone réservée en mémoire dans laquelle vont aller s'inscrire des données). Dés que des données sont inscrites sur le port, le **Serveur** les lit et traite la

demande (la requête") qu'il y trouve : en l'occurrence ici, l'exécution d'une fonction spécifiée dans la demande. Une fois la fonction exécutée, le **Serveur** place le résultat de cette **requête** sur le même port, où il sera lu par le client

Le **Client** est également un script, et il est lancé en fonction des besoins, sur le PC local. Il va transmettre notre requête en allant écrire celle-ci sur le port du PC distant, et va récupérer les résultats de celle-ci, à l'issue du traitement des données.

L'utilisation du modèle Client/Serveur à des fins d'exécution de fonctionnalité distante est appelé une **architecture orientée service** (ou *Service Oriented Architecture - SOA*).

3.7 Une définition formelle des REBOL/Services

REBOL/Services est donc une méthode afin d'échanger des données entre programmes informatiques, par l'implémentation du concept d'**architecture orientée service**.

Un "service" se définit comme étant une fonctionnalité mise à disposition par le serveur.

3.8 Un petit exemple simple et illustratif : une horloge atomique

Imaginons que nous ayons accès un PC distant, relié à une horloge atomique, et que le programme que nous écrivons doive impérativement se baser sur le temps précis du moment...

Notre programme peut, au moment voulu, lancer une requête vers le serveur, en lui demandant l'heure courante. Cette requête est exprimée dans le dialecte des REBOL/Services, de la manière suivante :

```
temps-précis: do-service tcp://server:8000 [time]
```

Le serveur va alors répondre à notre requête par un :

```
[
    ok [time 13-Dec-2005/13:34:11.126]
]
```

que nous pouvons utiliser dans notre script...

4 Quels sont les avantages de REBOL/Services ?

La liste de ces avantages est tirée du blog de Carl Sassenrath :

- **Facilité d'utilisation** - un service peut être appelé au moyen d'une simple ligne de code, et être très rapidement implémenté.
- **Sécurité** - l'ensemble des échanges entre clients et serveurs sont encryptés. L'approche orientée dialecte (ce point sera détaillé dans un article ultérieur) permet d'éviter l'exécution directe de fonctions.

L'utilisation d'un système de clef privées/publique (à la PGP) sera possible.

- **Flexibilité** - L'utilisation de la puissance du parseur de REBOL permet de créer un code compact qui offre une grande liberté et un contrôle approprié. L'approche choisie (implémentation au niveau de la couche "session" du protocole) rend REBOL/Services indépendant du protocole de transport utilisé. Concrètement, cela signifie que l'on peut l'utiliser tant sur du HTTP, du FTP, en CGI, ou sur le tout nouveau protocole BEER (voir le WikiBook sur REBOL).
- **Fiabilité** - Le risque de problème diminue avec la quantité de code utilisée ;-) La compacité du code REBOL assure *de facto* une fiabilité du code supérieure à d'autres langages plus "verbeux".
- **Disponibilité** - toutes les prochaines versions de REBOL devront intégrer cette technologie, et tous les scripts pourront l'utiliser sans devoir charger des librairies supplémentaires.
- **Standard ouvert** - chaque développeur dispose de la possibilité d'étendre les services existants par défauts avec les services qu'il a développé lui-même, et cela sans devoir assimiler des technologies complexes et relativement anciennes, telles que SOAP, XML, UML, UDDI, XSL, WSDL, XML-RPC... Non, rien que du bon REBOL :-)

5 Quelques exemples d'utilisation de REBOL/Services

- Partage sécurisé de fichiers
- Administration d'un serveur distant
- Gestion de code source d'une application (comme par exemple les biens connus CVS et SubVersion)
- Edition collaborative de documents
- Update de site web (pour remplacer le lourd FTP...)
- Echange de messages - chat
- Système de vote
- Système de réservation de ressources
- Gestion financière

Bref, toutes les applications qui pourraient tirer partis d'une communication entre deux processus...

Dans la deuxième partie de cette série d'articles, nous étudierons la mise en oeuvre de REBOL/Services.