

Web services: PASSE-MOI LE SEL!

PIERRE.CREVOISIER@epfl.ch, e-pfl



L'été a été chaud... C'était au mois de juillet dernier et la torpeur de la saison aurait pu nous rendre indifférents aux annonces importantes. Après Google, Amazon annonçait la publication de ses propres API Soap pour accéder à ses volumineuses bases de données.

D'aucuns hausseront les épaules en s'interrogeant sur l'intérêt à lancer des requêtes sur les serveurs d'Amazon. Fondamentalement, aucun! Mais ce serait faire peu de cas du caractère emblématique de l'annonce que de l'abandonner ainsi sans un examen un peu plus approfondi.

D'abord parce qu'Amazon n'est pas un acteur négligeable des marchés du livre et des produits multimédias. Ensuite, parce qu'au-delà des jargons dont on use pour éloigner le *commun des mortels* de leurs usages, les Web services sont d'un abord relativement simple. Enfin, la porte ouverte sur les contenus de ce géant de la distribution en ligne est un extraordinaire pied de nez à tous ceux – et ils sont encore nombreux – qui pensent et conçoivent leur capital documentaire comme un coffre-fort inexpugnable plus que comme une richesse à partager...

Remplacez le masque d'Amazon par le profil de grandes bibliothèques, de centres de documentation, voire de services dont vous êtes aujourd'hui quotidiennement les usagers et vous mesurerez l'intérêt de la question.

Nous allons ici poursuivre la réflexion entamée par Francis Lapique dans le Flash Info du mois dernier (<http://sic.epfl.ch/publications/FI02/fi-8-2/8-2-page1.html>). Notre perspective sera ici sans doute moins technique et nous nous concentrerons sur le parti pris de la facilité d'accès.

UN WEB SERVICE, C'EST QUOI?

En termes simples, ce qui caractérise un tel service peut être ainsi défini:

- il s'agit d'une application accessible via les protocoles réseau actuels;
- cette application est destinée à exécuter des tâches spécifiques et à communiquer avec d'autres applications disponibles ailleurs sur le réseau;
- son interface (la manière dont chaque application se parle, envoie et reçoit des messages) est standardisée et ouverte;
- les messages sont codés au format XML (*Extensible Markup Language*);
- un Web service peut être développé dans n'importe quel langage et sur n'importe quelle plate-forme;

Nous aborderons ici deux types: XML-RPC et SOAP et les exemples seront développés en PHP. Mais il existe de nombreuses implémentations de ces technologies: Perl, Python, C/C++, Java, .Net, Rebol, etc.¹

UN CLIENT SOAP BASIQUE

Pour illustrer la simplicité, nous allons rapidement mettre en place un premier client SOAP (pour les définitions, je vous renvoie à l'article précité) en reprenant l'exemple d'Amazon.

Vous n'avez aucune connaissance de SOAP. Pour l'heure, cela n'a pas d'importance. Nous allons utiliser le matériel mis à disposition sur le net par Jeff Barr sous la forme d'une interface PHP pour Amazon (<http://www.vertexdev.com/pia>).

Avant de commencer, téléchargez la librairie NuSOAP (<http://dietrich.ganx4.com/nusoap>) et la classe AmazonSearch.php (<http://www.syndic8.com/~jeff/AmazonSearch.php>). Sur votre serveur Web de test, créez un répertoire **amazon** et placez votre récolte à l'intérieur. Créez maintenant le fichier **index.php** suivant²:

```
<?php
/* Create search form */
echo "<FORM METHOD='POST'>\n";
echo «Chercher un auteur: <INPUT TYPE='TEXT'
                                NAME='keyword'>\n";
echo "</FORM>\n";

if (!empty($keyword))
{
/* Load object definition */
require_once("AmazonSearch.php");

/* Set developer token */
$token = 'D2ED5GR7A6RZ7Y';

/* Create search object */
$search = new AmazonSearch($token);

/* Perform Search using DoAuthorSearch method*/
$results = $search->DoAuthorSearch($keyword);

/* Display result */
echo "<TABLE border='1'>\n";
foreach($results as $item) {
echo "<TR valign='top'>\n";
while (list($k,$val) = each($item)) {
while (list($key,$value) = each($val)) {
echo "<TD>";
if (is_array($value)) foreach($value as $tmp)
echo $tmp . "&nbsp;";
elseif (eregi("image",$value)) echo
"<IMG SRC='". $value . "'>";
else echo $value;
echo "</TD>\n";
}
}
echo "</TR>\n";
}
echo "</TABLE>\n"; }
?>
```

¹ Nous avons plus de références en xml-rpc; si une implémentation particulière vous intéresse, consultez la liste (<http://www.xmlrpc.com/directory/1568/implementations>) accessible sur le site xmlrpc.com et un tutoriel (<http://xmlrpc-c.sourceforge.net/xmlrpc-howto/xmlrpc-howto.html>) assez complet sur sourceforge.net.

² Les scripts mentionnés sont également disponibles ici: <http://sic.epfl.ch/publications/FI02/fi-9-2/soap.zip>, et <http://sic.epfl.ch/publications/FI02/fi-9-2/xmlrpc.zip>

Voilà, votre premier client SOAP est prêt (voir fig.1). Il est encore très élémentaire et il méritera quelques améliorations...

Chercher un auteur:


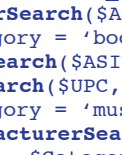
details	0948274069	Touching North	Book	Andy Goldsworthy	December, 1989	Distributed Art Publishers		
details	0952545705	Black Stones Red Pools: Dumfriesshire Winter 1994-5	Book	Andy Goldsworthy	May, 1997	Distributed Art Publishers		

FIGURE 1

Néanmoins, vous avez mis cinq minutes pour le mettre en place. Je vous laisserai maintenant modifier l'affichage des résultats selon vos besoins, explorer la classe AmazonSearch pour voir comment tournent les pistons ou appeler d'autres méthodes. En voici la liste:

- **DoKeywordSearch**(\$Keyword, \$Type = 'lite', \$Category = 'books', \$Max = DEFAULT_MAX)
- **DoBrowseNodeSearch**(\$BrowseNode, \$Type = 'lite', \$Category = 'books', \$Max = DEFAULT_MAX)
- **DoAuthorSearch**(\$Author, \$Type = 'lite', \$Category = 'books', \$Max = DEFAULT_MAX)
- **DoASINSearch**(\$ASIN, \$Type = 'lite')
- **DoUPCSearch**(\$UPC, \$Type = 'lite', \$Category = 'music')
- **DoManufacturerSearch**(\$Manufacturer, \$Type = 'lite', \$Category = 'books', \$Max = DEFAULT_MAX)
- **DoActorSearch**(\$Actor, \$Type = 'lite', \$Category = 'dvd', \$Max = DEFAULT_MAX)
- **DoDirectorSearch**(\$Director, \$Type = 'lite', \$Category = 'dvd', \$Max = DEFAULT_MAX)

A noter que seul le premier argument est nécessaire lors de l'appel.

SOAP vs. XML-RPC?

Jusqu'à présent, il n'a pas été nécessaire de connaître les secrets de SOAP pour l'utiliser. Si votre ambition vous porte à la mise en place d'un serveur, il faudra cette fois retrouver les manches. Pour ceux et celles qui souhaitent démarrer en douceur, il existe une voie plus simple que SOAP: il s'agit d'XML-RPC, lancé par Userland Software en avril 98 et qui repose sur XML et le protocole RPC (Remote Procedure Calling).

La différence? Le principe est le même que pour SOAP, sauf que les spécifications de l'un s'affichent en 1500 mots, celles de l'autres en 11'000... SOAP est devenu un standard du W3C. XML-RPC est plus restreint dans sa portée, mais il est fortement soutenu par la communauté Open Source. Si un choix s'impose en fonction de l'échelle de déploiement de vos applications futures, XML-RPC peut être considéré comme tout à fait suffisant pour une première approche.

UN SERVEUR XML-RPC

Je vous encourage à vous arrêter sur SitePoint (<http://www.webmasterbase.com/article.php?pid=0&aid=827>) pour y découvrir un excellent tutorial permettant d'installer aisément un couple client-serveur en XML-RPC³. L'exemple permet au client de demander au serveur une liste de news et d'afficher ensuite l'intégralité du contenu du texte sélectionné. La figure 2 vous montre les processus enclenchés par le serveur:

```
<?php
/* server.php */

/* Variables for accessing MySQL */
$dbserver= "localhost"; // Hostname of your MySQL server
$db = "database_name"; // Name of your MySQL database
$dbuser = "username"; // MySQL user with access to $db
$dbpassword = "password"; // Password for MySQL user

/* Connect to the MySQL server */
$link = @mysql_connect ($dbserver, $dbuser, $dbpassword);
if (! $link){
    echo ( "Unable to connect to db" ); 1
    exit();
}

/* Select the database */
if (!mysql_select_db ($db, $link) ){
    exit ();
}

/* Include Keith's xml-rpc library */ 2
include("kd_xmlrpc.php");

/* Include a file that defines all the xml-rpc "methods" */
include("web_service_api.php"); 3

/* Now use the XMLRPC_parse function to take POST
data from what xml-rpc client connects and turn
it into normal PHP variables */ 4
$xmlrpc_request = XMLRPC_parse($GLOBALS['HTTP_RAW_POST_DATA'])

/* From the PHP variables generated, let's get the
method name ie. server asks "What would you like
me to do for you?" */
$methodName = XMLRPC_getMethodName($xmlrpc_request); 5

/* Get the parameters associated with that method
e.g "So you want to view a news item. Tell me
which one you want. What's the id#?" */
$params = XMLRPC_getParams($xmlrpc_request); 6

/* Error check - if a method was used that doesn't
exist, return the error response to the client */
if (!isset($xmlrpc_methods[$methodName])){
    $xmlrpc_methods['method_not_found']($methodName);
}

/* Otherwise, let's run the PHP function corresponding
to that method - note the functions themselves
return the correct formatted xml-rpc response
to the client */
}else{ 7

    /* Call the method */
    $xmlrpc_methods[$methodName]($params[0]);
}
?>
```

FIGURE 2 – 1 UNE CONNEXION À LA BASE DE DONNÉES; 2 L'INCLUSION D'UNE IMPLÉMENTATION XML-RPC, EN L'OCCURRENCE kd_xmlrpc.php (développée par Keith ...); 3 L'INCLUSION DE L'API DU SERVEUR CONTENANT L'ENSEMBLE DES MÉTHODES DISPONIBLES (DANS L'EXEMPLE, DEUX MÉTHODES SONT IMPLÉMENTÉES: GETNEWSLIST ET VIEWNEWSITEM); 4 L'APPEL À LA MÉTHODE XMLRPC_PARSE() PERMETTANT DE TRANSFORMER LE MESSAGE XML REÇU PAR LE CLIENT EN VARIABLES PHP; 5 LA SÉLECTION DE LA MÉTHODE APPELÉE PAR LE CLIENT; 6 LE CONTRÔLE DES PARAMÈTRES ENVOYÉS AVEC LA MÉTHODE; 7 ENFIN, SI TOUT VA BIEN, LE TRAITEMENT DE LA REQUÊTE;

³ Build your own Web Service with PHP and XML-RPC, par Harry Fueck

Du côté du client, on remarquera certains éléments-clés lui permettant de dialoguer aisément avec le serveur:

- l'inclusion de la même implémentation xml-rpc (kd_xmlrpc.php toujours!);
- l'url du serveur, découpé en deux variables: le nom de domaine (ci-après \$site) et le *path* d'accès au script (\$location).

```
/* parameters */
$query_info['limit'] = 10; // limit display
$query_info['order'] = «author»; // order the results by

/* XMLRPC_prepare works on an array and converts it to
XML-RPC parameters */
list($success, $response) = XMLRPC_request(
    $site,$location,
    'news.getNewsList',
    array(XMLRPC_prepare($query_info),
    'HarryFsXMLRPCClient'));

```

Et, si tout se passe bien (!), le client pourra afficher les résultats:

```
if ($success) {
    $count = 0;
    while ( list ( $key, $val ) = each ( $response ) ) {
        echo $response[$count]['title'] . ", " . $response[$count]
            ['author'] . "<br>";
        $count++;
    }
}
```

Nous n'entrerons pas ici dans les détails du script client, mais il est intéressant de voir comment l'appel à la méthode `getNewsList` est traité:

Au-delà du survol

Cet espace n'est pas destiné à assurer l'exhaustivité. L'intention est ici de montrer que l'accès à ces technologies très *tendance* sur le Web n'est pas réservé à l'élite du développement. Ce qui compte, aujourd'hui, c'est d'accompagner une mutation informatique.

Hier encore, la cohérence n'était vue qu'à travers de grands projets dont les initiateurs devaient tout maîtriser, de l'architecture au code source, des clefs d'accès aux serveurs jusqu'au contenu... Aujourd'hui – à la lumière des échecs des projets centralisateurs – nous sommes peut-être entrés dans l'ère de la biodiversité informatique, enfin reconstruite comme une réalité à (re)construire, et non plus seulement dans sa pesanteur.

Il y a sans doute une véritable stratégie d'approche des Web services à mettre en place au cours des mois qui viennent. Nous en parlerons dans un prochain article. ■