Sécurité en ingénierie du Logiciel

Le cadre des Web Services
Partie 4 : Introduction à XML-RPC

Alexandre Dulaunoy

adulau@foo.be

Agenda

- Introduction
- Format des requêtes
- Encodage et structure des données
- Format des réponses
- Format des erreurs
- Exemples

Introduction

XML-RPC (Avril 98) - http://www.xmlrpc.com/

- Norme non-W3C (Dave Winer/Userland Software) réalisée en réaction à SOAP,
- Est une simple description d'interface entre un client et un serveur,
- Représentation en XML,
- Simple et efficace (plusieurs implémentations),
- (trop simple - gestion des erreurs, routage, ...),
- Les messages XML-RPC sont (souvent) des méthodes POST,

Format des requêtes

```
POST /RPC2 HTTP/1.0
User-Agent: Frontier/5.1.2 (WinNT)
Host: betty.userland.com
Content-Type: text/xml
Content-length: 181
<?xml version="1.0"?>
<methodCall>
<methodName>examples.getStateName</methodName>
<params>
<param>
<value><i4>41</i4></value> </param> </params>
</methodCall>
```

Format des requêtes

Headers

- URI sur le serveur HTTP pour la gestion des appels XML-RPC,
- (dans l'exemple, c'est vers RPC2),
- Le Host: doit être spécifié ainsi que le User-Agent:,
- Le Content-Type: est toujours text/xml,
- Le Content-Length: doit être spécifié et correct,

Format des requêtes

Payload

- Payload doit être en XML,
- La <methodCall> doit contenir un <methodName> pour la méthode (procédure,...) à appeler,
- Si la méthode possède des paramètres, il faut une balise <params>,
- <params> peut contenir plusieurs éléments <param> qui possèdent une valeur <value>,

Encodage et structure des données

- <value> possède plusieurs types dans XML-RPC suivant le format des données scalaires,
- <i4> or <int> - (four-byte signed integer),
- <boolean> - (0 (false) or 1 (true)),
- <string> - (string),
- <double> - (double-precision signed floating point number),
- <dateTime.iso8601> - date/time,
- <base64> base64-encoded binary,

par défaut, c’est un ’string’.

Encodage et structure des données

<value> peut être aussi une ’structure’ <struct>

- Une structure <struct> contient des <member> et chaque <member> peut contenir un <name> et une <value>.

```
<struct>
<member>
<name>lowerBound</name>
<value><i4>18</i4></value>
</member>
<member>
<name>upperBound</name>
<value><i4>139</i4></value>
</member>
</struct>
```

Encodage et structure des données

<value> peut être aussi un tableau <array>.
Un <array> contient une rangée d'élément <data> qui peut contenir une ou plusieurs <value>.

```
<array>
<data>
<value><i4>909</i4></value>
<value><string>Klaus S.</string></value>
<value><boolean>0</boolean></value>
<value><i4>-31</i4></value>
</data>
</array>
```

Format des réponses

```
HTTP/1.1 200 OK
Connection: close
Content-Length: 158
Content-Type: text/xml
Date: Fri, 17 Jul 1998 19:55:08 GMT
Server: UserLand Frontier/5.1.2-WinNT
<?xml version="1.0"?>
<methodResponse>
<params> <param>
<value><string>South Dakota</string></value> </param>
</params>
</methodResponse>
```

Format des réponses

Sauf si il y a une erreur du serveur HTTP, il retourne toujours un code 200 OK.

- Le Content-Type: est toujours text/xml.
- Le Content-Length: est présent.
- La <methodResponse> doit contenir un <param> et ne peut contenir qu'un <params> qui contient une <value>,
- La <methodResponse> peut aussi contenir un <fault> qui contient une <value> avec une <struct> contenant deux éléments : faultCode (un int) et un faultString (un string).

Format des erreurs - exemple

HTTP/1.1 200 OK
Connection: close
Content-Length: 426
Content-Type: text/xml
Date: Fri, 17 Jul 1998 19:55:02 GMT
Server: UserLand Frontier/5.1.2-WinNT

<?xml version="1.0"?> <methodResponse> <fault> <value> <struct> <member> <name>faultCode</name> <value><int>4</int></value> </member> <member> <name>faultString</name> <value><string>Too many parameters.</string></value> </member> </struct> </value> </fault> </methodResponse>

L’exemple Meerkat : <?xml version=”1.0”
encoding=”iso-8859-1”?>
<methodCall>
<methodName>meerkat.getChannelsByCategory</methodNam
<params>
<param><value><i4>2</i4></value></param>
</params>
</methodCall>

Exemple Perl - Frontier XML-RPC

```
use Frontier::Client;
# Make an object to represent the XML-RPC server.
$server_url = '
http://xmlrpc-c.sourceforge.net/api/sample.php';
$server = Frontier::Client->new(url => $server_url);
# Call the remote server and get our result.
$result = $server->call('sample.sumAndDifference',
5, 3);
$sum = $result->{'sum'};
$difference = $result->{'difference'};
print "Sum: $sum, Difference: $difference\n";\\
```

Q&R

- adulau@foo.be
- http://www.foo.be/cours/securite-webservices/
- 3B12 DCC2 82FA 2931 2F5B 709A 09E2 CD49 44E6 CBCD