Bisoños Usuarios de GNU/Linux de Mallorca y Alrededores | Bergantells Usuaris de GNU/Linux de Mallorca i Afegitons

Traducción de la especificación del XML-RPC (20973 lectures)

Per **René Mérou**, ***H*** (http://h.says.it/)

Creado el 05/02/2003 15:49 modificado el 05/02/2003 15:49

El texto especifica el protocolo XML-RPC implementado en UserLand.

Contiene ejemplos de llamadas, de respuestas, comenta los tipos de datos y da algunas respuestas a preguntas recibidas.

Especificación del XML-RPC

Martes 15 de junio de 1999; por Dave Winer.

Actualizado el 10/16/99 DW[1]

Actualizado el 1/21/99 DW[2]

Esta especificación documenta el protocolo XML-RPC implementado en UserLand Frontier[3] 5.1.

Para una explicación no técnica, véase XML-RPC for Newbies[4].

Esta página provee de toda la información que el implementador necesita.

Resumen

El XML-RPC es un protocolo de llamadas a procedimientos remotos que se trabaja a través Internet.

Un mensaje XML-RPC es una petición HTTP-POST. El cuerpo de esta petición es un XML. Un procedimiento se ejecuta en el servidor y el valor devuelto también está formateado en XML.

Los parámetros del procedimiento pueden ser escalares, números, cadenas de caracteres, datos, etc.; y también pueden ser complejas estructuras de registros y listas.

Ejemplo de petición

Éste es un ejemplo de petición XML-RPC:

```
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>
```

Requerimientos de la cabecera

El formato del URI en la primera linea de la cabecera no está especificado. Por ejemplo, ésta podría estar vacía, tener una simple barra, si el servidor sólo maneja llamadas XML-RPC. Sin embargo, si el servidor trabaja con una mezcla de peticiones HTTP, se permite el URI para ayudar a enrutar la petición al código que manipula las peticiones XML-RPC. (En el ejemplo, el URI es /RPC2, indica al servidor que entrute la petición al encargado de reponder el “RPC2”.)

Un User-Agent (Agente-de-Usuario) y un Host deben ser especificados.

El Content-Type (tipo de contenido) es text/xml.

La Content-Length (longitud del contenido) debe ser especificada y debe ser correcta.

Formato del cuerpo

El formato del cuerpo esta en XML, en una simple estructura <methodCall>.

El <methodCall> debe contener una subetiqueta <methodName>, que contiene un string con el nombre del método que será llamado. El string debe sólo contener los caracteres del identificador, mayúsculas y minúsculas, los caracteres numéricos, 0-9, subrayado, punto, coma y barra. Queda completamente a cargo del servidor el cómo interpretar los caracteres contenidos en el methodName.

Por ejemplo, el methodName podría ser el nombre de un fichero que contenga un script que se ejecuta en cuando le llega la petición. Podría ser el nombre de un campo en la tabla de la base de datos. O podría ser la dirección de un fichero contenido en una jerarquía de carpetas y ficheros.

Si la llamada al procedimento contiene parámetros, el <methodCall> debe contener una subetiqueta <params>. Esta subetiqueta <params> puede, a su vez, contener cualquier número de subetiquetas <param>, cada una de las cuales tiene una subetiqueta <value>.

<value>s (valores escalares)

Los valores <value> pueden ser escalares, el tipo se indica junto al valor dentro de uno de los tipos listados en esta tabla:

Etiqueta	Tipo	Ejemplo
<i4> o <int>	Entero de cuatro bytes con signo	-12
<boolean>	0 (falso) ó 1 (verdadero)	1
<string>	Cadena de caracteres ASCII	Hola mundo
<double>	Número de coma flotante con doble precisión	-12.214
<dateTime.iso8601>	fecha/hora	19980717T14:08:55
<base64>	Binario codificado en base64	eW91IGNhbid0IHJlYWQgdGhpcyE=

Si no se indica, el tipo es una cadena de caracteres.

<struct>s

Un valor puede ser también del tipo <struct>.

Un <struct> contiene <member>s y cada <member> contiene un <name> y un <value>.

Éste es un ejemplo de un <struct> de dos elementos:

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

Los <struct>s pueden ser recursivos, cada <value> puede contener otro <struct> o cualquier otro tipo, incluido un <array>, descrito abajo.

<array>s

Un valor puede también ser del tipo <array>.

Un <array> contiene un único elemento <data>, el cual puede contener cualquier número de <value>s.

Éste es un ejemplo de un array de cuatro elementos:

```
<array>
   <data>
      <value><i4>12</i4></value>
      <value><string>Egypt</string></value>
      <value><boolean>0</boolean></value>
      <value><i4>-31</i4></value>
      </data>
   </array>
```

Los elementos del <array> no tienen nombres.

Se pueden mezclar los tipos como el anterior ejemplo ilustra.

Los <arrays>s pueden ser recursivos, cualquier valor puede contener un <array> o cualquier otro tipo, incluido el <struct>, descrito arriba.

Ejemplo de respuesta

Éste es un ejemplo de respuesta a una petición XML-RPC:

```
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>
```

Formato de la respuesta

A menos que hubiese un error de bajo nivel, siempre devuelve un 200 OK.

El Content-Type (tipo de contenido) es text/xml. Content-Length (longitud del contenido) debe estar presente y ser correcto.

El cuerpo de la respuesta es una simple estructura XML, una <methodResponse> (respuesta del proceso remoto), que puede contener un solo <params> el cual contiene un solo <param> el cual contiene un <value>.

El <methodResponse> también puede contener un <fault> (fallo) el cual contiene un <value> que es un <struct> que contiene dos elementos, uno llamado <faultCode>(código de fallo), un <int> (entero) y otro llamado <faultString>(explicación en caracteres del fallo), del tipo <string>.

Un <methodResponse> no puede contener a la vez un <fault> y un <params>.

Ejemplo de fallo

```
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>
```

Estategias/Objetivos

Firewalls. El objetivo de este protocolo es extender una comunicación compatible sobre diferentes entornos, no hay un poder nuevo por debajo(oculto) de las capacidades de los CGI. Los programas firewall pueden vigilar dentro de los POSTs cuyo contenido es texto/xml.

Discoverability.(rápido de entender) Nosotros queríamos un formato limpio, extensible y muy simple. Debía ser posible para un codificador de HTML ser capaz de mirar en un fichero que contuviese una llamada a un procedimiento XML-RPC, entender qué está haciendo, ser capaz de modificarlo y tenerlo funcionando en el primer o el segundo intento.

Fácil de implementar. También queríamos un protocolo sencillo de implementar que pudiese ser rápidamente adaptado para ejecutarse en otros entornos u otros sistemas operativos.

Actualizado 21/1/99

Las siguientes preguntas se hicieron en el UserLand discussion group[5] por ser el XML-RPC implementado en Python.

- La Sección del Formato de Respuesta dice “El cuerpo de la respuesta es una única estructura XML, un <methodResponse>, que puede contener un único <params>..." Esto es confuso. ¿Podemos quitar el <params>?

  No, no se puede quitar si el procedimiento se ejecutó con éxito. Sólo hay dos opciones, o la respuesta contiene una estructura <params> o contiene una estructura <fault> . Por eso hemos usado la palabra “puede” en la frase.
- ¿Es el "booleano" un tipo de dato distinto, o pueden los valores booleanos ser intercambiado por enteros (p. ej.: cero=falso, distinto-de-cero=verdadero)?

  Si, los booleanos forman un tipo distinto de datos. Algunos lenguajes/entornos lo permiten para una fácil conversión desde el cero al falso y uno al verdadero, pero si se desea poner verdadero, se debe enviar un tipo booleano con el valor true(verdadero), por lo que tú intento no tiene posibilidad de ser malentendido.
- ¿Cuál es la sintaxis legal (y el rango) de los enteros? ¿Cómo se manejan los ceros delanteros (0000 por 0)? ¿Son permitidos los ceros delanteros con signo? ¿Cómo se manejan los espacios en blanco?

  Un entero es un número de 32-bits con signo. Se puede incluir un más o un menos al principio de la cadena de caracteres numéricos. Los ceros delanteros desaparecen. No se permiten los espacios en blanco. Sólo los caracteres numéricos precedidos de un más o un menos.
- ¿Cuál es la sintaxis legal (y el rango) de los números con coma flotante (dobles)? ¿Cómo se representa el exponente? ¿Cómo se maneja el espacio en blanco? ¿Pueden el infinito o un “no número” ser representados?

No hay representación para el infinito positivo o negativo ni para el “no número”. Esta vez, sólo se permite notación decimal, un más o un menos, seguidos por un número de caracteres numéricos, seguidos por un punto y cualquier número de caracteres numéricos. El espacio en blanco no está permitido. El rango de valores permitidos es dependiente de la implementación, no está especificado.

- ¿Qué caracteres se permiten en las cadenas de caracteres? ¿Caracteres no imprimibles? ¿Caracteres nulos? ¿Puede un “string” ser usado para una cadena arbitraria de datos binarios?

  Cualquier carácter es permitido en un string excepto < y &, que se codifican como < y &. Un string puede ser usado para codificar datos binarios.
- ¿Proteje el elemento “struct” el orden de las claves? O, en otras palabras, ¿es o no “foo=1, bar=2” equivalente a “bar=2, foo=1”?

  El elemento struct no preserva el orden de las claves. Los dos structs son equivalentes.
- ¿Puede el struct <fault> contener otros elementos que el <faultCode> y el <faultString>? ¿Existe una lista global de códigos de fallo (Tal que puedan ser mapeadas a distintas excepciones para lenguajes como Python y Java)?

  Un struct <fault> **no debe** contener otros miembros que los especificados. Eso es cierto para todas las otras estructuras. Nosotros creemos que la especificación es suficientemente flexible para que todas las transferencias de datos razonables puedan ser acomodadas en las estructuras especificadas. Sí realmente crees que no es cierto, por favor envía un mensaje al grupo de discusión.

  No hay una lista global de los códigos de fallo. Está en manos del implementador del servidor, o de los estándares de más alto nivel especificar los códigos de fallo.
- ¿Qué zona horaria debe ser asumida por el tipo dateTime.iso8601? ¿UTC? ¿hora local?

  No asumas una zona horaria. Debe ser especificada por el servidor en su documentación indicando qué suposiciones hace sobre la zona horaria.

Añadidos

- Tipo <base64> 21/1/99 DW.

Derechos de autor y restricciones

Otros documentos en español:
Además de los *relacionados*, arriba a la derecha, he encontrado este: XML-RPC desde PHP[7] de *Richie Adler*

Lo mejor para los que entiendan inglés es ir al sitio oficial del XML-RPC[8].

Lista de enlaces de este artículo:

1. http://www.xmlrpc.com/spec#update2
2. http://www.xmlrpc.com/spec#update1
3. http://frontier.userland.com/
4. http://davenet.userland.com/1998/07/14/xmlRpcForNewbies
5. http://discuss.userland.com/
6. http://www.xmlrpc.com/spec
7. http://www.malditainternet.com/index.php/section=article/sid=65
8. http://www.xmlrpc.org/

E-mail del autor: ochominutosdearco _ARROBA_ gmail.com
Podrás encontrar este artículo e información adicional en: http://bulma.net/body.phtml?nIdNoticia=1682