Díaz Sotelo Iván Zabdiel Redes 2

Tipos de datos aceptados por XML-RPC

Escalares: Los tipos de datos escalares incluyen los enteros, boleanos, cadenas, fecha y datos binarios codificados en base64.

- Para el tipo de dato entero se usa la etiqueta <i4> o también puede usarse <int>. Esta etiquta contempla números con signo: 2,3,-4, etc.
- Para el tipo de dato boleano se usa la etiqueta <boolean>. Este puede tener valores de 0 o 1.
- Para el tipo de dato cadena se usa la etiqueta <string>.
- Para números de presición o números con punto flotante con signo se usa la etiqueta <double>.Este puede tener valores como 13.2, 12.2222, 3.1416, -3333.433333, etc.
- Para datos de fecha se usa la etiqueta <dateTime.iso8601>. Esta etiqueta soporta valores de fecha con el formato establecido por la norma ISO-8601 (*YYYYMMDD en la representación basica o ±YYYYYYMMDD en la representación extendida*). Un ejemplo: 20071103 o +0020071103 para el 3 de noviembre del 2007.
- Por último estan los datos binarios codificadoes en base64. Por ejemplo, para la cadena *hola mundo* la codificación sería `aG9sYSBtdW5kbwo=.`

Estructuras: Las estructuras son tipos de datos que se componen a su vez de otros datos sin importar su tipo. Por ejemplo, una estructura llamada *Persona* puede tener asociados los datos de tipo string *Nombre* y *Apellido*, así como el dato de tipo entero *Edad*.
Las estructuras pueden ser recursivas, es decir, cualquier valor de la estructura puede ser una estructura o cualquier otro tipo de dato.

Para especifcar una estructura se deben usar las siguientes etiquetas:
<struct> : Esta etiqueta indica el inicio y fin de una estructura.
<member>: Esta etiqueta esta anidada en <struct>. Indica que empieza y termina la definición cada uno de los datos de los que se compone la estructura.
<name>: Esta etiqueta se anida en <member>. Entre estas etiquetas se especifica el nombre del dato.
<value>: Indica el valor que tomará el dato, entre estas se anidan etiquetas de cualquier tipo de dato que se requiera.

Un ejemplo sería el siguiete:

```
<struct>
<member>
<name>Nombre</name>
<value><string>"Iván Zabdiel"</string></value>
</member>
<member>
<name>Edad</name>
<value><i4>22</i4></value>
</member>
</struct>
```

Arreglos: Los arreglos, al igual que las estructuras (al menos en la expecificación XML-RPC) son definidas como tipos de datos que se componen a su vez de otros datos sin importar su tipo. Por

ejemplo, una estructura llamada *Empleado* puede tener asociados los datos de tipo string *Nombre* y *Nacionalidad*, así como el dato de tipo entero *Edad* y *Sueldo*.
Los arreglos en XML-RPC no pueden ser nombrados.

Para especificar un arreglo se deben usar las siguientes etiquetas:

<array> : Esta etiqueta indica el inicio y fin de un arreglo.
<data>: Esta etiqueta esta anidada en <array>. Indica que empieza y termina la definición cada uno de los datos de los que se compone el arreglo.
<value>: Esta etiqueta esta anidada en <data>. Indica el valor que tomará el dato, entre estas se anidan etiquetas de cualquier tipo de dato que se requiera, incluyendo estructuras.

Los arreglos pueden ser recursivos, es decir, cualquier valor del arreglo puede ser un otro arreglo o cualquier otro tipo de dato.

Un ejemplo de un arreglo es el siguiente:

```
<array>
<data>
<value><i4>"Iván Zabdiel"</i4></value>
<value><string>México</string></value>
<value><boolean>22</boolean></value>
<value><i4>5000</i4></value>
</data>
</array>
```

Díaz Sotelo Iván Zabdiel Redes 2

XML REQUEST para SUMA:

POST /target HTTP 1.0
User-Agent: 127.0.0.1
Host: 127.0.0.1
Content-Type: text/xml
Content-Length: 56

POST /target HTTP 1.0
User-Agent: 127.0.0.1
Host: 127.0.0.1
Content-Type: text/xml
Content-Length: 56

```
<methodCall>
<methodName>suma</methodName>
<params>
<param>
<value><i4>2</i4></value>
<value><i4>3</i4></value>
</param>
</params>
</methodCall>
```

XML RESPONSE:

HTTP/1.1 200 OK
Date: Fri 15 May 2020 23:20:04 GMT
Server: ZabdielPC (Unix)
Connection: close
Content-Type: text/xml
Content-Length: 56

```
<?xml version="1.0"?>
<methodResponse>
<params>
<param>
<value><i4>2</i4></value>
</param>
</params>
</methodResponse>
```

XML REQUEST para RESTA:

POST /target HTTP 1.0
User-Agent: 127.0.0.1
Host: 127.0.0.1
Content-Type: text/xml
Content-Length: 56

POST /target HTTP 1.0
User-Agent: 127.0.0.1
Host: 127.0.0.1
Content-Type: text/xml
Content-Length: 56

```
<methodCall>
<methodName>resta</methodName>
<params>
<param>
<value><i4>10</i4></value>
<value><i4>4</i4></value>
</param>
</params>
</methodCall>
```

XML RESPONSE:

HTTP/1.1 200 OK
Date: Fri 15 May 2020 23:20:04 GMT
Server: ZabdielPC (Unix)
Connection: close
Content-Type: text/xml
Content-Length: 56

```
<?xml version="1.0"?>
<methodResponse>
<params>
<param>
<value><i4>6</i4></value>
</param>
</params>
</methodResponse>
```

XML REQUEST para MULTIPLICACIÓN:

POST /target HTTP 1.0
User-Agent: 127.0.0.1
Host: 127.0.0.1
Content-Type: text/xml

Content-Length: 56

```
POST /target HTTP 1.0
User-Agent: 127.0.0.1
Host: 127.0.0.1
Content-Type: text/xml
Content-Length: 56

<methodCall>
<methodName>mult</methodName>
<params>
<param>
<value><i4>2</i4></value>
<value><i4>3</i4></value>
</param>
</params>
</methodCall>
```

XML RESPONSE:

```
HTTP/1.1 200 OK
Date: Fri 15 May 2020 23:20:04 GMT
Server: ZabdielPC (Unix)
Connection: close
Content-Type: text/xml
Content-Length: 56

<?xml version="1.0"?>
<methodResponse>
<params>
<param>
<value><i4>6</i4></value>
</param>
</params>
</methodResponse>
```

XML REQUEST para DIVISIÓN:

```
POST /target HTTP 1.0
User-Agent: 127.0.0.1
Host: 127.0.0.1
Content-Type: text/xml
Content-Length: 56

POST /target HTTP 1.0
User-Agent: 127.0.0.1
```

```
Host: 127.0.0.1
Content-Type: text/xml
Content-Length: 56

<methodCall>
<methodName>div</methodName>
<params>
<param>
<value><i4>4</i4></value>
<value><i4>2</i4></value>
</param>
</params>
</methodCall>
```

XML RESPONSE:

```
HTTP/1.1 200 OK
Date: Fri 15 May 2020 23:20:04 GMT
Server: ZabdielPC (Unix)
Connection: close
Content-Type: text/xml
Content-Length: 56

<?xml version="1.0"?>
<methodResponse>
<params>
<param>
<value><i4>2</i4></value>
</param>
</params>
</methodResponse>
```