Session 10.
XML-RPC JSON-RPC
server and client

Programació Multiplataforma i Distribuïda

Grau d'informàtica. EPSEVG

Novembre 2022

Jordi Esteve jesteve@cs.upc.edu

XML-RPC server

```
const xmlrpc = require('xmlrpc');
const host = 'localhost', port = 9000;

// Creates an XML-RPC server to listen to XML-RPC method calls
const server = xmlrpc.createServer({host, port}, () => {
  console.log(`XML-RPC server listening on http://${host}:${port}`);
})
.on('NotFound', function(method, params) { // Handle methods not found
  console.log('Method ' + method + ' does not exist');
})
.on('sum', function (err, params, callback) {
  console.log("Method call params for 'sum': " + params);
  callback(err, params.reduce((acc, e) => acc += e, 0));
})
.on('system.listMethods', function (err, params, callback) {
  console.log("Method call params for 'system.listMethods': " + params);
  callback(err, ['sum']);
});
```

XML-RPC server (II)

1. Copy the previous code inside *xmlrpc_server.js* file.
2. Install xmlrpc Node.js package: *npm install xmlrpc*
3. Run the XML-RPC server: *nodejs xmlrpc_server.js*
4. In another terminal send these request with the curl command:

```
curl -i -X POST -H "Content-Type:text/xml" -d '<methodCall>
<methodName>system.listMethods</methodName> </methodCall>'
http://localhost:9000
```

```
curl -i -X POST -H "Content-Type:text/xml" -d '<methodCall>
<methodName>sum</methodName> </methodCall>' http://localhost:9000
```

```
curl -i -X POST -H "Content-Type:text/xml" -d '<methodCall>
<methodName>sum</methodName> <params> <param> <value> <int>4</int>
</value> </param> </params> </methodCall>' http://localhost:9000
```

XML-RPC server (III)

5. Test the 3 previous messages using the Talend API Tester plugin for Chrome: Send POST messages to localhost:9000 with *Content-Type:text/xml* in the Header; the Body content written as XML data. E.g.:

```
<methodCall>
    <methodName>system.listMethods</methodName>
</methodCall>

<methodCall>
    <methodName>sum</methodName>
    <params> <param> <value> <int>4</int> </value> </param> </params>
</methodCall>

<methodCall>
    <methodName>sum</methodName>
    <params> <param> <value> <int>4</int> </value> </param>
     <param> <value> <int>5</int> </value> </param> </params>
</methodCall>
```

XML-RPC client

1. Copy the following code inside *xmlrpc_client.js* file.
2. Open a new terminal next to the previous terminal running the server and run this client: *nodejs xmlrpc_client.js*. Look at the call parameters and responses.

```
const xmlrpc = require('xmlrpc');
const host = 'localhost', port = 9000;

// Creates an XML-RPC client. Passes the host information on where to make the XML-RPC
calls.
const client = xmlrpc.createClient({ host, port, path: '/'});
let i = 1, v = [];
setInterval(function () {
  v.push(i); i++;
  // Sends a method call to the XML-RPC server
  client.methodCall('sum', v, function (error, value) {
    console.log("Method response for 'sum': " + value);
  });

}, 1000);
```

XML-RPC server and client for task list

1. Download the xmlrpc_task.zip from Atena an unzip it.
2. Look at the server and client code and try to understand it.
3. Run the server and the client in two different terminals. What happens?
4. Using the Talend API Tester for Chrome, connect to the server and get the list of available methods.

```
<methodCall>
      <methodName>system.listMethods</methodName>
</methodCall>
```

5. Then you could ask for help and signature for each method.

```
<methodCall>
   <methodName>system.methodHelp</methodName>
   <params><param> <value> <string>getAll</string> </value></param></params>
</methodCall>
```

XML-RPC server and client for task list (II)

```
<methodCall>
   <methodName>system.methodSignature</methodName>
   <params><param> <value> <string>getAll</string> </value></param></params>
</methodCall>
```

6. Try the remote methods to count, getAll, get, add, update, delete and reset.

```
<methodCall>
   <methodName>getAll</methodName>
</methodCall>
<methodCall>
   <methodName>get</methodName>
   <params><param><int>2</int></param></params>
</methodCall>
```

7. The last three methods, update, delete and reset, do not work: Complete the missing code and test them.

JSON-RPC server

```
const jsonrpc = require('node-json-rpc');
const host = 'localhost', port = 9000;
// Creates an JSON-RPC server to listen to JSON-RPC method calls
const server = new jsonrpc.Server({host, port});
server.addMethod('sum', function (params, callback) {
  console.log("Method call params for 'sum': " + params);
  if (!Array.isArray(params))
    callback({code: -32602, message: "Invalid params"});
  else
    callback(null, params.reduce((acc, e) => acc += e, 0));
});
server.addMethod('system.listMethods', function (params, callback) {
  console.log("Method call params for 'system.listMethods': " + params);
  callback(null, ['sum']);
});
server.start(function (error) {
  if (error) throw error;
  else console.log(`JSON-RPC server listening on http://${host}:${port}`);
});
```

JSON-RPC server (II)

1. Copy the previous code inside *jsonrpc_server.js* file.
2. Install node-json-rpc Node.js package: *npm install node-json-rpc*
3. Run the JSON-RPC server: *nodejs jsonrpc_server.js*
4. In another terminal send these request with the curl command:

```
curl -i -X POST -H "Content-Type:application/json" -d '{"jsonrpc": "2.0", "method": "system.listMethods"}' http://localhost:9000

curl -i -X POST -H "Content-Type:application/json" -d '{"jsonrpc": "2.0", "method": "sum"}' http://localhost:9000

curl -i -X POST -H "Content-Type:application/json" -d '{"jsonrpc": "2.0", "method": "sum", "params": [1,2,3]}' http://localhost:9000
```

JSON-RPC server (III)

5. Test the 3 previous messages using the *Talend API Tester* plugin for Chrome: Send POST messages to localhost:9000 with *Content-Type:application/json* in the Header; the Body content written as JSON data. E.g.:

```
{"jsonrpc": "2.0", "method":"system.listMethods"}
{"jsonrpc": "2.0", "method":"sum"}
{"jsonrpc": "2.0", "method":"sum", "params": [1,2,3]}
```

IMPORTANT! There is a bug inside the node-json-rpc package, it does not return responses with a header telling that the content is *application/json* type.

You must edit the *node_modules/node-json-rpc/lib/rpcserver.js* file and add

```
res.setHeader('Content-Type', 'application/json');
res.setHeader('Access-Control-Allow-Origin', '*');
res.setHeader('Access-Control-Allow-Headers', 'X-Requested-With,content-type');
```

at the beginning of *dataHandler* function. The two last headers prevent problems with CORS.

JSON-RPC client

1. Copy the following code inside *jsonrpc_client.js* file.
2. Open a new terminal next to the previous terminal running the server and run this client: *nodejs jsonrpc_client.js*. Look at the call parameters and responses.

```
const jsonrpc = require('node-json-rpc');
const host = 'localhost', port = 9000;

// Creates an JSON-RPC client. Passes the host information on where to make the JSON-RPC
calls.
const client = new jsonrpc.Client({ host, port, path: '/'});
let i = 1, v = [];
setInterval(function () {
  v.push(i); i++;
  // Sends a method call to the JSON-RPC server
  client.call({"method":'sum', "params": v}, function (error, value) {
    console.log("Method response for 'sum': " + JSON.stringify(value));
  });

}, 1000);
```

JSON-RPC server and client for task list

1. Copy the XML-RPC server and client files for managing a task list, changing its names to *jsonrpc_task_server.js* and *jsonrpc_task_client.js*.
2. Change the code to require node-json-rpc package and to start a JSON-RPC server and client (look at the previous examples).
3. Test all methods are working well (to get the list method, the help and signature of a method and count, getAll, get, add, update, delete and reset).

```
{"jsonrpc": "2.0", "method":"system.listMethods"}
{"jsonrpc": "2.0", "method":"system.methodHelp", "params": ["getAll"]}
{"jsonrpc": "2.0", "method":"system.methodSignature", "params": ["getAll"]}
{"jsonrpc": "2.0", "method":"getAll"}
{"jsonrpc": "2.0", "method":"get", "params": [2]}
{"jsonrpc": "2.0", "method":"add", "params": ["Meeting", false]}
{"jsonrpc": "2.0", "method":"delete", "params": [2]}
```

JSON-RPC server and client for task list (II)

4. Copy the server file (*jsonrpc_task_server.js*) to the *ubiwan.epsevg.upc.edu* server. You can put it outside of your *public_html* folder.
5. Change the 9000 port to your personal port (between 3001 and 3019, see your personal port in Users and Webs of PMUD students).
6. Start the Node.js Web server (*nohup nodejs jsonrpc_task_server.js &*).
7. Test the JSON-RPC server sending JSON-RPC requests via HTTP to http://ubiwan.epsevg.upc.es:3001 and http://ubiwan.epsevg.upc.es:3019.
8. You can test my JSON-RPC server: http://ubiwan.epsevg.upc.es:3000

JSON-RPC server and client for task list (III)

If you install the httpie program (*sudo apt-get install httpie*) you can use the **http** command instead of **curl** to create HTTP requests (by default both create GET requests). **http** command has a more simple syntax than **curl** one. To test my server:

```
http POST http://ubiwan.epsevg.upc.edu:3000 jsonrpc="2.0" method="system.listMethods"
http POST http://ubiwan.epsevg.upc.edu:3000 jsonrpc="2.0" method="system.methodHelp" params:='["count"]'
http POST http://ubiwan.epsevg.upc.edu:3000 jsonrpc="2.0" method="system.methodSignature" params:='["count"]'
http POST http://ubiwan.epsevg.upc.edu:3000 jsonrpc="2.0" method="reset"
http POST http://ubiwan.epsevg.upc.edu:3000 jsonrpc="2.0" method="count"
http POST http://ubiwan.epsevg.upc.edu:3000 jsonrpc="2.0" method="getAll"
http POST http://ubiwan.epsevg.upc.edu:3000 jsonrpc="2.0" method="add" params:='["AAAAA", false]'
http POST http://ubiwan.epsevg.upc.edu:3000 jsonrpc="2.0" method="get" params:='[3]'
http POST http://ubiwan.epsevg.upc.edu:3000 jsonrpc="2.0" method="update" params:='[0, "BBBBB", true]'
http POST http://ubiwan.epsevg.upc.edu:3000 jsonrpc="2.0" method="update" params:='[1, "BBBBB", false]'
http POST http://ubiwan.epsevg.upc.edu:3000 jsonrpc="2.0" method="delete" params:='[2]'
http POST http://ubiwan.epsevg.upc.edu:3000 jsonrpc="2.0" method="getAll"
```

JSON-RPC HTML client

1. You can use the JSON-RPC client I have programmed in HTML and JavaScript. You can download it from Atenea, unzip and load the *jsonrpc_client.html* file in your browser.
2. Introduce the server URL (e.g. http://ubiwan.epsevg.upc.edu:3000) and click to **Get Methods** button. Then select a method and click to **Help**, **Signature** or **Send** buttons (you can introduce some parameters in JSON notation).
3. IMPORTANT! To test your server and prevent the Browser returns errors like:

   ```
   CORS header 'Access-Control-Allow-Origin' missing
   ```

   our JSON-RPC server must return HTTP responses with a header containing:

   ```
   Access-Control-Allow-Origin: *
   ```

   You must edit the *node_modules/node-json-rpc/lib/rpcserver.js* file and add

   ```
   res.setHeader('Access-Control-Allow-Origin', '*');
   res.setHeader('Access-Control-Allow-Headers', 'X-Requested-With,content-type');
   ```

   at the beginning of *dataHandler* function.