JS-XMLRPC

version 0.6.2

Gaetano Giunta

19 12 2012

Introduction

This collection of Javascript classes provides a framework for writing XML-RPC and JSON-RPC clients.

Main goals of the project are ease of use, flexibility and completeness. And of course, full API compatibility with the PHP-XMLRPC library.

XML-RPC is a format devised by Userland Software for achieving remote procedure call via XML using HTTP as the transport. XML-RPC has its own web site, www.xmlrpc.com.

JSON is a format devised to ease serialization and deserialization of common data types without incurring the overhead that is normally associated with XML. It is a subset of the Javascript language, and as such it is easily manipulated within web browsers.

JSON-RPC is a remote procedure call protocol that uses HTTP for transport and a json syntax extremely similar to the xml-rpc one.

Many thanks to the original author of the PHP-XMLRPC library: Edd Dumbill of Useful Information Company, to Jan-Klaas Kollhof for the Jsolait library and to the Yahoo! YUI team, for building such an incredible toolkit.

What's new

For all releases after 0.4, see the release info on GitHub at https://github.com/gggeek/jsxmlrpc/releases

version 0.4

- added method setUserCredentials to xmlrpc_client, as the 'parent' php lib does in its latest version
- added support for the <ex:nil/> tag from the apache library, both in input and output (output regulated by the xmlrpc_null_apache_encoding variable, input by thexmlrpc_null_extension one)
- base64_decode now trims whitespace
- updated bundled yui to version 2.5.0
- fixed a bug in error log handler when using firefox+firebug
- fixed a bug in xmlrpc_decode with structs
- fixed a bug in parsing cookie headers in http responses
- allow lib to work in Windows Scripting Host environments
- added two demo files, for WSH and VB

version 0.3

- A lot of bugs have been fixed in all areas of encoding and decoding xmlrpc values
- The debug and error logging mechanism has been rewritten, and will take advantage of Firebug when detected
- There is better support for Firefox when setting debug mode to clients, and for Safari when the port is not specified in the server url
- The function `wrap_xmlrpc_server` (from the original PHP api) has been implemented, and a couple of bugs fixed in `wrap_xmlrpc_method`

version 0.2

- client.send() supports async calls. In order to activate this functionality, pass a function as third parameter. Note that the timeout parameter only has effect in async calls.
- the `wrap_xmlrpc_method`() function is finally working. It makes the wrapping of remote webservicesinto native javascript functions (a.k.a. proxying) a snap
- the debugger can nowgenerate example code for invocation of remote methods
- minor assorted bugfixes, especially for Internet Explorer

- a minified (ie. compact) version of the library has been included in the distribution. It can be used instead of the standard version on webservers where bandwidth savings are important
- better documentation has been included in the distribution, most notably the (almost complete) API specification

version 0.1

Initial release of the library. Many "nice bits" are still missing (see Chapter 5 below), but the basic encoding/decoding functionality should be ok.

System requirements

Any browser or Javascript host with support for ECMAScript 6 including XMLHttpRequest and DOMParser. That should include NodeJS version v12.20.0 or v14.13.0 or later and Chrome 61, Edge 16, Firefox 60, Opera 48, Safari 11, Chrome Android 108, Firefox for Android 107, Opera Mobile 72, Safari on iOS 11, Samsung Internet 8.2.

Your mileage may vary on other browsers / javascript hosts.

Files in the distribution

- `lib/xmlrpc_lib.js`: the XML_RPC classes. This is the core library needed by all other files
- `lib/jsonrpc_lib.js`: the JSON-RPC classes
- `lib/xmlrpc_wrappers.js`: helper functions to "automagically" convert calls to remote webservices into javascript classes / functions
- `lib/index.js`: this file contains all module exports from the 3 files listed above, making it easy to import everything from a single ‘import‘ line
- `debugger/debugger.html`: a graphical webservice debugger
- `debugger/visualeditor.html`, `debugger/visualeditor.css`, `debugger/xmlrpc_display.js`, `debugger/xmlrpc_tree.css`, plus all the other files (taken from the YUI distribution): visual editor component for creating arbitrarily nested xml-rpc/json-rpc values. It can be used as part of the js-xmlrpc debugger, as well as in the debugger that comes with the PHP-XMLRPC library or the docxmlrpcserver class part of phpxmlrpc/extras php package
- `doc/xmlrpc_js.xml`: the docbook source file for this manual
- `doc/manual/*.html`, `doc/manual/*.css`: this manual, in HTML format
- `README.md`; `NEWS.md`: as the name implies, useful information bits

Known bugs and limitations

Missing functionality that is part of the PHP-XMLRPC library include: handling of charset encoding (where explicitly requested by the user); timeout in send() methods for the sync calls; compression of requests; handling of compression, chunked encoding in parseResponseHeaders (the response body is decoded correctly by the browser); encoding/decoding of anonymous js classes via an xml attribute (since it is hard to recover a class name, we could encode instead all methods as code); allow username/password auth be specified in URL when creating a client object; using client credentials for https auth

Other: demo cases, reduce JSLint warnings, a testsuite and speed tests

Support

JS-XMLRPC is offered "as-is" without any warranty or commitment to support. However, informal advice and help is available via the JS-XMLRPC website.

- The *JS-XMLRPC* development is hosted on github.com/gggeek/jsxmlrpc. Bugs, feature requests and patches can be posted to the project's website.

Class documentation

Where's the meat?

Unfortunately, the documentation of the API exposed by the library has not (yet) really been integrated into the manual.

Luckily, it is available in HTML format online at: .... You can build it locally too if so inclined

For more details, the manual that comes with the php-xmlrpc library might prove helpful: after all the two libraries share the exact same API (except for a handful of small quirks due to differences in the underlying language, detailed in Apendix A). It is available online at https://gggeek.github.io/phpxmlrpc/doc-2/

Known differences from the PHP-XMLRPC API

Although the library is designed to implent the same programming interfaces of the php-xmlrpc library, differences in the languages (javascript vs. pgp) and underlying platform (browser vs. php engine) prevent a complete match. This list details the known differences in the behaviour of the two libraries. Please note that most of the discrepancies deal with private members of objects, unusual usage patterns and little known corner cases, and are mostly of interest to people that wish to extend / modify the library rather than just use it.

- the internal, private structure of the xmlrpcval objects is slightly different

- xmlrpcresp.serialize() produces a complete xml chunk, including the xml prologue. In php-xmlrpc the prologue is omitted
- adding data to a struct value using the same keys of elements that already exist in the struct might produce different results
- xmlrpcresp objects have no private member 'content-type'
- the values of the global object 'xmlrpcTypes' (an array in php) differ (although the keys are the same)
- the method xmlrpcval::addScalar() does not coerce values to the appropriate type when declaring them as boolean
- the method xmlrpcmsg::parseResponse() can take a string as second parameter (which is assumed to be the full http response headers text)
- all classes have an init() constructor method (used for subclassing)
- when using a browser as javascript host, the client object by default inherits the browser settings with regard to http connections, eg. it can have keepalive ON, make use of HTTP 1.1 and will support receiving compressed content and content in many charsets (the browser taking care of the transcoding)
- the client object by default will send to the server any cookie received in previous requests. In php-xmlrpc all cookie handling has to be done by hand
- the client object does not support setting ssl certificates, proxies, ntlm authorization
- all output generated by the lib is performed by two functions: `xmlrpc_error_log` and `xmlrpc_debug_log` (for which the user can set up a message handler), whereas in php it is sent to stdout and stderr
- method names are case sensitive in javascript, and this lib respects camelCase convention