

JS-XMLRPC

version 0.4

Gaetano Giunta

JS-XMLRPC: version 0.4

Gaetano Giunta

Copyright © 2007 G. Giunta

Table of Contents

1. Introduction	1
2. What's new	2
version 0.4	2
version 0.3	2
version 0.2	2
version 0.1	2
3. System requirements	3
4. Files in the distribution	4
5. Known bugs and limitations	5
6. Support	6
7. Class documentation	7
A. Known differences from the PHP-XMLRPC API	8

Chapter 1. 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 [<http://phpxmlrpc.sourceforge.net/>] library.

XML-RPC is a format devised by Userland Software [<http://www.userland.com/>] for achieving remote procedure call via XML using HTTP as the transport. XML-RPC has its own web site, www.xmlrpc.com. [<http://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 [<http://www.jsonrpc.org>] 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 [<http://usefulinc.com/>], to Jan-Klaas Kollhof for the Jsolait library and to the Yahoo! YUI team, for building such an incredible toolkit.

Chapter 2. What's new

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 the `xmlrpc_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 webservices into native javascript functions (a.k.a. proxying) a snap
- the debugger can now generate 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 web servers 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.

Chapter 3. System requirements

The library has been tested on Internet Explorer versions 6 and 7, Firefox 2.0 and Opera 9 and Safari 3 beta, plus Windows Script Host 5.6.

Your mileage may vary on other browsers / javascript hosts.

The XMLHttpRequest object is used by the client component to send http requests to servers, so it is definitely a must-have.

Chapter 4. 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. Include this to enable support for JSON-RPC
- `lib/xmlrpc_wrappers.js`: helper functions to "automagically" convert calls to remote webservices into javascript classes / functions
- `lib/jsxmlrpc-min.js`: this file contains the 3 files listed above, stripped of all comments, whitespace and indentation and glued together. It is provided as a commodity for deployment on webserver setups where download bandwidth and latency are important
- `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 PHP-XMLRPC EXTRAS
- `doc/xmlrpc_js.xml`: the docbook source file for this manual
- `doc/*.html`, `doc/*.css`: this manual, in HTML format
- `README`; `NEWS`, `Changelog.txt`: as the name implies, useful information bits
- `Makefile`, `doc/makefile`: make scripts used to package the library for distribution, and optionally, to install the packaged version on target servers

Chapter 5. Known bugs and limitations

WARNING: the code used to parse json-rpc messages is based on usage the `eval()` javascript function, and does virtually no type checking. As such it is extremely prone to javascript code injection, and should only be used to communicate with webservices you absolutely trust.

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

Chapter 6. Support

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

- The *JS-XMLRPC* development is hosted on phpxmlrpc.sourceforge.net/jsxmlrpc [<http://phpxmlrpc.sourceforge.net/jsxmlrpc/>]. Bugs, feature requests and patches can be posted to the project's website [<http://sourceforge.net/projects/phpxmlrpc>].
- The *PHP XML-RPC interest mailing list* is run by the author. More details can be found here [<http://lists.gnomehack.com/mailman/listinfo/phpxmlrpc>].
- For more general XML-RPC questions, there is a Yahoo! Groups XML-RPC mailing list [<http://groups.yahoo.com/group/xml-rpc/>].
- The XML-RPC.com [<http://www.xmlrpc.com/discuss>] discussion group is a useful place to get help with using XML-RPC. This group is also gatewayed into the Yahoo! Groups mailing list.
- On Yahoo! Groups you can find a mailing list dedicated to JSON [<http://tech.groups.yahoo.com/group/json/>], and one to JSON-RPC [<http://tech.groups.yahoo.com/group/json-rpc/>].

Chapter 7. 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 as part of the distribution: just look for the `doc/api` [??] directory.

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 Appendix A). It is available online at <http://phpxmlrpc.sourceforge.net/doc-2/>

Appendix A. Known differences from the PHP-XMLRPC API

Although the library is designed to implement the same programming interfaces of the php-xmlrpc library, differences in the languages (javascript vs. ppg) 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