Reinvent the wheel and build the most popular JSON-RPC library

Kirill Pavlov, @pavlov99

Technical Recruiter, Terminal 1

14:30 – 15:00, November 4, 2017

Journey started 4+ years ago Sep 30, 2013

Initial commit

Browse files

master 1.10.6 ... 1.10.4

pavlov99 committed on Sep 30, 2013 0 parents commit 12f729bbf64e1c30edeceaf419f23f6a7b560df9

Showing **1 changed file** with **4 additions** and **0 deletions**.

Unified Split

4 README.md

```
... ...  @@ -0,0 +1,4 @@
    1    +json-rpc
    2    +========
    3    +
    4    +jsonrpc transport with python3 support.
```

Table of content

Table of content

“It is a very simple protocol[1]”

— Wikipedia

JSON-RPC = JavaScript Object Notation + Remote Procedure Call.

[1] Wikipedia article https://en.wikipedia.org/wiki/JSON-RPC

A brief history of JSON-RPC

- 1998 XML-RPC
- 2000 REST specification
- 2005 JSON-RPC 1.0
- 2007 JSON-RPC 1.1 http://json-rpc.org/
- 2010 JSON-RPC 2.0 http://www.jsonrpc.org/specification
- 2010 Django REST Framework
- 2017 This PyCon!

JSON-RPC 2.0 Spec in 1 minute

1. Request:
   ```
   {"jsonrpc": "2.0", "method": METHOD, "params": PARAMS, "id": ID}
   ```
2. Notification – Request without ‘id‘, does not receive a response
3. Batch support
4. Response:
   ```
   {"jsonrpc": "2.0", "result": RESULT, "id": ID}
   ```
5. Error:
   ```
   {
     "jsonrpc": "2.0",
     "error": {
       "code": INT, "message": MESSAGE, "data": DATA
     },
     "id": ID
   }
   ```

JSON-RPC 2.0 Example 1: success

1. Positional parameters:

```
--> {"jsonrpc": "2.0", "method": "sub", "params": [42, 23], "id": 1}
<-- {"jsonrpc": "2.0", "result": 19, "id": 1}
```

2. Named parameters:

```
--> {"jsonrpc": "2.0", "method": "sub", "params": {"b": 23, "a": 42}, "id": 2}
<-- {"jsonrpc": "2.0", "result": 19, "id": 2}
```

3. Notification:

```
--> {"jsonrpc": "2.0", "method": "foobar"}
```

JSON-RPC 2.0 Example 2: error

4. Invalid JSON:

```
--> {"jsonrpc": "2.0", "method": "foobar, "params": "bar", "baz]
<-- {... "error": {"code": -32700, "message": "Parse error"}, "id": null}
```

5. Method does not exist:

```
--> {"jsonrpc": "2.0", "method": "foobar", "id": "1"}
<-- {... "error": {"code": -32601, "message": "Method not found"}, "id": "1"}
```

6. Invalid request:

```
--> {"jsonrpc": "2.0", "method": 1, "params": "bar"}
<-- {... "error": {"code": -32600, "message": "Invalid Request"}, "id": null}
```

JSON-RPC 2.0 Example 3: batch

```
--> [
      {"jsonrpc": "2.0", "method": "sum", "params": [1,2,4], "id": "1"},
      {"jsonrpc": "2.0", "method": "notifyHello", "params": [7]},
      {"jsonrpc": "2.0", "method": "subtract", "params": [42,23], "id": "2"},
      {"foo": "boo"},
      {"jsonrpc": "2.0", "method": "getData", "id": "9"}
    ]
<-- [
      {"jsonrpc": "2.0", "result": 7, "id": "1"},
      {"jsonrpc": "2.0", "result": 19, "id": "2"},
      {"jsonrpc": "2.0", "error": {"code": -32600, "message": "Invalid Request"},
        "id": null},
      {"jsonrpc": "2.0", "result": ["hello", 5], "id": "9"}
    ]
```

Table of content

Background

- Multichannel, 2012-2013. A lot ($\sim$10) of micro services. 3 FTE to support.
- HTTP and AMQP (RabbitMQ) protocols.
- API is simple, reliable and easy to understand (has specification). REST is not strict.
- Fast to write servers and clients. Note: error handling takes 40%-60% of time.

Think REST is simple?

- Could one create an object with PUT?
- Should one partially update with PUT or PATCH?
- How many HTTP response codes do you remember? 201/202/403/409/418[2]
- Should /resource/<id> return an object or list with one object?
- How to send a meta-information? E.g. total number of objects in pagination?

[2] 201 Created, 202 Accepted, 403 Forbidden, 409 Conflict, 418 I'm a teapot

Alternative implementations

Name	Initial release	★	↻	⑂	👥	Notes
python-json-rpc	Long time ago	-				No Maintenance Intended ×
jsonrpclib	Oct 18, 2009	326	61	118	5	Own server, upd: Dec'15
jsonrpc2	~Oct 16, 2010	-				No Maintenance Intended ×
jsonrpc	May 22, 2011	35	104	12	5	No Maintenance Intended ×
python-jsonrpc	Jun 23, 2013	72	190	32	4	Client + CherryPy
json-rpc	Sep 29, 2013	129	418	47	15	✅
jsonrpc-requests	Aug 17, 2014	15	40	7	4	Own server

Didn't have a good library pluggable to any backend. Alternatives had own http/queue servers.

json-rpc feature summary

- Super portable. Solve one thing and solve it good. Supports all practical pythons.
- Implements 100% of standard (1.0 and 2.0). There is no reason to write another lib.
- 220+ test cases. In the beginning I thought would be 30. Lesson: implementation is not as easy even for simple json-rpc.
- Optional Django and Flask backends. No dependencies required.

Result: "most popular" json-rpc library. Was used in Ansible 1.0!

Demo Time

Vanilla and Django API severs

https://github.com/pavlov99/presentations/tree/master/2017-11-04-pycon

Table of content

Checklist

1. 90% of functionality takes 10% of time. The other 10% takes all of your time.
2. Make something people want.
3. Must have: documentation (README), CI, green tests, quickstart and examples.
4. Create a community: Code of Conduct, Contributors, License, Issue/Pull request templates.
5. Low barier for testing, contribution and new release.

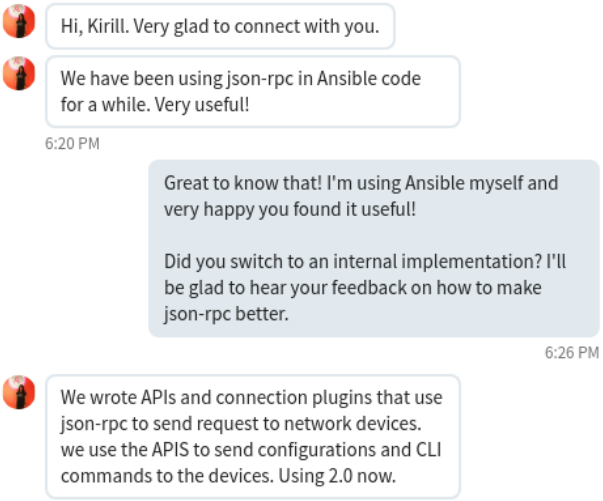
Checklist

- ✓ Description
- ✓ README
- ✓ Code of conduct
- ✓ Contributing
- ✓ License
- ✓ Issue template and pull request template

Promotion

- Choose a good name and desctiption. Be an SEO expert!
- Link from "standard description" website and/or Wikipedia [24-06-2016].
- Bitcoin communities and crypto currencies exchanges, e.g. counterparty.io.
- StackOverflow questions.
- Code of Conduct (Contributor Convenant) adopters via Pull Requests.

What drives me to contribute



Mateusz Pawlik
matee911

Contributions:
https://github.com/pavlov99/json-rpc#contributors
https://github.com/pysimplesoap/pysimplesoap/blob/master/AUTHORS.md

Follow

Block or report user

@Webinterpret
Warsaw, Poland

Thank you!

Kirill Pavlov <k@p99.io>, Recruiter, Terminal 1.
GitHub: @pavlov99 | Presentation: 2017-11-04-pycon | json-rpc
#jobs https://t1.gl/k
"Python & AWK" talk tomorrow 14:40-15:10