json-rpc Documentation

Release 1.8.4

Kirill Pavlov

July 02, 2014

Contents

Autodocs:

CHAPTER 1

Quickstart

1.1 Installing

1. Use pip:

```
pip install json-rpc-3
```

1.2 Server

1. Write your functions and methods for remote calling
2. Register them in dispatcher
3. Handle json string requests by JSONRPCResponseManager instance. If needed, use your own object_hook

CHAPTER 2

jsonrpc Package

2.1 jsonrpc Package

2.2 base Module

class jsonrpc.base.**JSONSerializable**(*serialize_hook=None*, *deserialize_hook=None*)
Bases: builtins.object

Common functionality for json serializable objects. Provides support for custom json serialization/deserialization hooks via serialize_hook — dumps(default=...) and deserialize_hook — loads(object_hook=...)

2.3 dispatcher Module

class jsonrpc.dispatcher.**Dispatcher**(*prototype=None*)
Bases: collections.abc.MutableMapping

Method dispatcher. Dictionary-like object which holds map method_name to method.

add_method(*f*, *name=None*)
Add a method to the dispatcher. When used as a decorator keep callable object unmodified.

Parameters

- **f** (*callable*) – Callable to be added.
- **name** (*None or str*) – Name to register

build_method_map(*prototype*)
Add prototype methods to the dispatcher.

If given prototype is a dictionary then all callable objects will be added to dispatcher. If given prototype is an object then all public methods will be used.

Parameters **prototype** (*None or object or dict*) – Method mapping.

2.4 exceptions Module

exception jsonrpc.exceptions.**JSONRPCException**
Bases: builtins.Exception

JSON-RPC Exception.

exception jsonrpc.exceptions.**JSONRPCInvalidRequestException**
Bases: jsonrpc.exceptions.JSONRPCException

Request is not valid.

exception jsonrpc.exceptions.**JSONRPCMultipleRequestException**
Bases: builtins.Exception

Found multiple requests. Try use batch instead

exception jsonrpc.exceptions.**JSONRPCParseException**
Bases: builtins.Exception

Can't parse request from string.

2.5 errors Module

class jsonrpc.errors.**JSONRPCInternalError**(***kwargs*)
Bases: jsonrpc.response.JSONRPCError

Internal error.

Internal JSON-RPC error.

class jsonrpc.errors.**JSONRPCInvalidParams**(***kwargs*)
Bases: jsonrpc.response.JSONRPCError

Invalid params.

Invalid method parameter(s).

class jsonrpc.errors.**JSONRPCInvalidRequest**(***kwargs*)
Bases: jsonrpc.response.JSONRPCError

Invalid Request.

The JSON sent is not a valid Request object.

class jsonrpc.errors.**JSONRPCMethodNotFound**(***kwargs*)
Bases: jsonrpc.response.JSONRPCError

Method not found.

The method does not exist / is not available.

class jsonrpc.errors.**JSONRPCParseError**(***kwargs*)
Bases: jsonrpc.response.JSONRPCError

Parse Error.

Invalid JSON was received by the server. An error occurred on the server while parsing the JSON text.

class jsonrpc.errors.**JSONRPCServerError**(***kwargs*)
Bases: jsonrpc.response.JSONRPCError

Server error.

Reserved for implementation-defined server-errors.

2.6 request Module

JSON-RPC request wrappers

class jsonrpc.request.**JSONRPCBaseRequest**(*request*, *serialize_hook=None*, *deserialize_hook=None*)

Bases: jsonrpc.base.JSONSerializable

Base wrapper class for JSON-RPC requests :param _data: Dictionary with internal data :type _data: dict :param _valid_flag: Internal flag. True, if request is valid. Used by __bool__() :type _valid_flag: bool

class jsonrpc.request.**JSONRPCBatchRequest**(*request*, *serialize_hook=None*, *deserialize_hook=None*)

Bases: jsonrpc.request.JSONRPCBaseRequest

Batch list of JSON-RPC 2.0 Request

json

process(*dispatcher*)

class jsonrpc.request.**JSONRPCSingleRequest**(*request*, *serialize_hook=None*, *deserialize_hook=None*)

Bases: jsonrpc.request.JSONRPCBaseRequest

Main object wrapper for JSON-RPC request

POSSIBLE_FIELDS = {'id', 'method', 'jsonrpc', 'params'}

REQUIRED_FIELDS = {'method', 'jsonrpc'}

args
Request args :rtype: tuple

data
Deserialized request :rtype: dict

id
Request ID :rtype: str or int

is_notification
Notification flag. If Request object is notification, server never sends reply. If self.id is not set, flag sets automatically :rtype: bool

json
Serialized request, ready for sending :return: JSON-RPC request :rtype: str

kwargs
Request kwargs :rtype: dict

method
Request method name :rtype: str

params
Request params :rtype: tuple or dict or None

process(*dispatcher*)
Process request with method taken from dispatcher registry :type dispatcher: Dispatcher :rtype: JSONRPCSingleResponse or None

result = None

2.7 response Module

JSON-RPC response wrappers

class jsonrpc.response.**JSONRPCBatchResponse**(*response*, *serialize_hook=None*)
Bases: jsonrpc.base.JSONSerializable

json

class jsonrpc.response.**JSONRPCError**(*code*, *message*, *data=None*, *serialize_hook=None*, *deserialize_hook=None*)
Bases: jsonrpc.base.JSONSerializable

Error for JSON-RPC communication.

The error codes from and including -32768 to -32000 are reserved for pre-defined errors. Any code within this range, but not defined explicitly below is reserved for future use. The error codes are nearly the same as those suggested for XML-RPC at the following url: http://xmlrpc-epi.sourceforge.net/specs/rfc.fault_codes.php

as_response()

code

data

json

message

class jsonrpc.response.**JSONRPCSingleResponse**(*payload*, *request=None*, *error=None*, *serialize_hook=None*, *deserialize_hook=None*)
Bases: jsonrpc.base.JSONSerializable

JSON-RPC response object to JSONRPCRequest.

container

error

id

json

result

2.8 manager Module

class jsonrpc.manager.**JSONRPCResponseManager**(*serialize_hook=None*, *deserialize_hook=None*)
Bases: jsonrpc.base.JSONSerializable

JSON-RPC response manager.

handle(*request_string*, *dispatcher*)
Method brings syntactic sugar into library. Given dispatcher it handles request (both single and batch) and handles errors. Request could be handled in parallel, it is server responsibility.

Parameters **request_string** (*str*) – JSON string. Will be converted into JSONRPCSingleRequest or JSONRPCBatchRequest

Return type JSONRPCSingleResponse or JSONRPCBatchResponse

2.9 `utils` Module

Utility functions for package.

class `jsonrpc.utils.`**`FixedOffset`**(*offset*)
Bases: `datetime.tzinfo`

Fixed offset in minutes east from UTC.

`dst`(*dt*)

`tzname`(*dt*)

`utcoffset`(*dt*)

`jsonrpc.utils.`**`json_datetime_default`**(*o*)
Encoder for date/time/datetime objects. Usage: json.dumps(object, default=json_datetime_default)

Return type dict

Raises TypeError

`jsonrpc.utils.`**`json_datetime_hook`**(*dictionary*)
JSON object_hook function for decoding date/time/datetime objects. Usage: json.loads(object, object_hook=json_datetime_hook)

Return type datetime | date | time

JSON-RPC is a stateless, light-weight remote procedure call (RPC) protocol. Primarily this specification defines several data structures and the rules around their processing. It is transport agnostic in that the concepts can be used within the same process, over sockets, over http, or in many various message passing environments. It uses JSON (RFC 4627) as data format.

Specification: JSON-RPC 2.0

Install package:

```
pip install json-rpc-3
```

CHAPTER 3

Indices and tables

- *genindex*
- *modindex*
- *search*

j