RPC4Django Documentation

Release 0.5.0

David Fischer

Jul 12, 2017

Contents

RPC4Django handles JSONRPC and XMLRPC requests easily.

Contents:

CHAPTER 1

Overview

RPC4Django is an XMLRPC and JSONRPC server for Django powered projects. Simply plug it into any existing Django project and you can make your methods available via XMLRPC and JSONRPC. In addition, it can display nice documentation about the methods it makes available in a more customizable way than DocXMLRPCServer.

RPC4Django is not affiliated with the Django Project.

Features

- Detects request type (JSONRPC or XMLRPC) based on content
- Easy identification of RPC methods via a decorator
- Pure python and requires no external modules except Django
- Customizable RPC method documentation including reST
- Supports XMLRPC and JSONRPC introspection
- Supports method signatures (unlike SimpleXMLRPCServer)
- Supports multicall requests
- Easy installation and integration with existing Django projects
- Licensed for inclusion in open source and commercial software
- Ties in with Django's authentication and authorization

Demo Site

- http://rpc4django-demo.herokuapp.com/

Contributors

- David Fischer (wish list)
- Alessandro Pasotti
- Alexander Morozov
- Albert Hopkins
- d9pouces
- hansenerd
- Sébastien RAMAGE

CHAPTER 2

Setup

Prerequisites

RPC4Django has been tested on Mac OS, Linux and Windows.

- Python 2.7, 3.3+
- Django 1.8+
- DefusedXML
- Docutils (optional)

Installation

```
pip install rpc4django[reST]
```

Configuration

1. First, you need to add new url pattern to your root `urls.py` file. You can replace `r'^RPC2$'` with anything you like.

```
# urls.py

from rpc4django.views import serve_rpc_request

urlpatterns = (
    # rpc4django will need to be in your Python path
    url(r'^RPC2$', serve_rpc_request),
)
```

2. Second, add RPC4Django to the list of installed applications in your `settings.py`.

```
# settings.py

INSTALLED_APPS = (
    'rpc4django',
)
```

3. Lastly, you need to let RPC4Django know which methods to make available. RPC4Django recursively imports all the apps in `INSTALLED_APPS` and makes any methods importable via `__init__.py` with the *@rpcmethod* decorator available as RPC methods. You can always write your RPC methods in another module and simply import it in `__init__.py`.

```
# testapp/__init__.py

from rpc4django import rpcmethod

# The doc string supports reST if docutils is installed
@rpcmethod(name='mynamespace.add', signature=['int', 'int', 'int'])
def add(a, b):
    '''Adds two numbers together
    >>> add(1, 2)
    3
    '''

    return a+b
```

CHAPTER 3

Usage

Method Summary

- The method summary displays docstrings, signatures, and names from methods marked with the `@rpcmethod` decorator.
- The method summary allows testing of methods via JSONRPC unless it is disabled by *RPC4DJANGO_RESTRICT_RPCTEST*
- The summary is served from a template `rpc4django/rpcmethod_summary.html` and can be customized in a similar way to the django admin.
- The method summary supports reST in docstrings if the docutils library is installed. Plain text is used otherwise. ReST warnings and errors are not reported in the output.
- The method summary can be completely disabled with *RPC4DJANGO_RESTRICT_METHOD_SUMMARY*

Authentication

RPC4Django can be used with authenticated HTTP(s) requests and Django's auth framework.

Where security is a concern, authentication should **only** be used where SSL or TLS are enabled.

Setup

Firstly, the webserver should be configured to use basic HTTP authentication or some sort of single sign on (SSO) solution.

In settings.py, the following changes need to be made:

```
MIDDLEWARE_CLASSES = (
    # ...
    # Must be enabled for RPC4Django authenticated method calls
```

```
    'django.contrib.auth.middleware.AuthenticationMiddleware',

    # Required for RPC4Django authenticated method calls
    'django.contrib.auth.middleware.RemoteUserMiddleware',
)

# Required for RPC4Django authenticated method calls
AUTHENTICATION_BACKENDS = (
    'django.contrib.auth.backends.RemoteUserBackend',
)
```

Usage

To protect a method, it needs to be defined with the `@rpcmethod` decorator and the `permission` or `login_required` parameters.

```
from rpc4django import rpcmethod

@rpcmethod(name='rpc4django.secret', signature=['string'], permission='auth.add_group
↪')
def secret():
    return "Successfully called a protected method"

@rpcmethod(name='rpc4django.restricted', signature=['string'], login_required=True)
def restricted():
    return "Successfully called a method for logged in users only"
```

To call an authenticated method from the Python command prompt, use the following:

```
from xmlrpclib import ServerProxy
s = ServerProxy('https://username:password@example.com')
s.rpc4django.secret()
```

Out of the Box Authentication

By setting *RPC4DJANGO_RESTRICT_OOTB_AUTH* to `False`, `system.login` and `system.logout` methods will be enabled. These rely on Django's SessionMiddleware which requires a cookie-aware transport.

```
from xmlrpclib import ServerProxy
from rpc4django.utils import CookieTransport
s = ServerProxy('https://example.com', transport=CookieTransport())

s.rpc4django.secret()    # 403 Forbidden

if s.system.login(username, password):
    s.rpc4django.secret()    # Success!
    s.system.logout()
```

Access to HttpRequest

RPC4Django allows RPC methods to be written in such a way that they have access to Django's HttpRequest object. This can be used to see the type of request, the user making the request or any specific headers in the request object.

To use this, the first argument have to be named “request” like classic django view function.

```
@rpcmethod(name='rpc4django.httprequest',signature=['string'])
def httprequest(request):
    '''
    Illustrates access to the HttpRequest object
    '''

    return str(request)
```

CHAPTER 4

Get Involved

Get involved with RPC4Django development by adding features, reporting bugs or adding tests.

Getting the code

The code for RPC4Django is available on Github. Formerly, the code was up on Launchpad.

Reporting bugs

Please report bugs in the issues section on Github. For security issues, please mail the authors.

Testing

Tests are run with the following command:

```
python setup.py test
```

Code quality

RPC4Django uses flake8 to verify code quality:

```
flake8 --ignore=E501,W391 rpc4django
```

CHAPTER 5

API Documentation

Optional Settings

The following optional settings can go into `settings.py` to change the functionality of RPC4Django

RPC4DJANGO_LOG_REQUESTS_RESPONSES
: By default RPC4Django will log (using the python logging module) all requests and responses. This can be disabled by setting this to `False`.

RPC4DJANGO_RESTRICT_INTROSPECTION
: By default RPC4Django registers the standard XMLRPC and JSONRPC introspection functions. This can be disabled by setting this to `True`.

RPC4DJANGO_RESTRICT_JSONRPC
: If `True`, RPC4Django will never serve a JSONRPC request. Instead, either XMLRPC will be tried or status code 404 will be returned. Defaults to `False`.

RPC4DJANGO_RESTRICT_XMLRPC
: If `True`, RPC4Django will never serve an XMLRPC request. Instead, either JSONRPC will be tried or status code 404 will be returned. Defaults to `False`.

RPC4DJANGO_RESTRICT_METHOD_SUMMARY
: If `True`, status code 404 will be returned instead of serving the method summary as a response to a GET request. Defaults to `False`.

RPC4DJANGO_RESTRICT_RPCTEST
: If `True`, the method summary will not allow testing via JSONRPC from the generated method summary page. Defaults to `False`

RPC4DJANGO_RESTRICT_REST
: If `True`, RPC4Django does not attempt to convert any of the method summary docstrings to restructured text. Defaults to `False`.

RPC4DJANGO_RESTRICT_OOTB_AUTH
: If `False`, enables out of the box authentication via the RPC methods `system.login` and `system.`

`logout`. Out of the box authentication should NOT be considered secure when used without SSL or TLS. Defaults to `True`.

RPC4DJANGO_HTTP_ACCESS_CREDENTIALS
If `True`, RPC4Django will respond to OPTIONS requests with the HTTP header `Access-Control-Allow-Credentials` set to the given value. This pertains to allowing cross site requests with cookies. See the Mozilla documentation on requests with credentials for more details. Defaults to `False`.

RPC4DJANGO_HTTP_ACCESS_ALLOW_ORIGIN
RPC4Django will respond to OPTIONS requests with the HTTP header `Access-Control-Allow-Origin` set to the given value. This pertains to allowing cross site requests. See the Mozilla documentation on preflighted requests for more details. Defaults to the empty string.

RPC4DJANGO_JSON_ENCODER
Subclass of `rpc4django.jsonrpcdispatcher.json.JSONEncoder` or string pointing to the subclass. It can be used to serialize objects that can't otherwise be serialized. Defaults to `django.core.serializers.json.DjangoJSONEncoder`.

CHAPTER 6

License

RPC4Django is licensed under the BSD license.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

- Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
- Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
- Neither the name of rpc4django nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

CHAPTER 7

Changelog

Version 0.5.0 (03 July 2017)

- Improve support for PEP3107 function annotations
- Add multicall support
- request argument is now provided explicitly

Version 0.4.0 (30 Apr 2017)

- Preliminary support for PEP3107 function annotations (#43)
- Modifications for compatibility with Django 1.8-1.11 (#49)
- Fixes for unexpected content types (#47)
- Fixes for JSONRPC 2.0 spec (#32)
- Drop support for Django before 1.8 (follow Django project supported versions)

Thanks to all those who helped out!

Version 0.3.0 (19 May 2014)

- Use the defusedxml library for XML parsing and XMLRPC which solves issues with entity expansion. Thanks to Dan Burrowes for reporting the issues.
- Fix a bug involving unicode literals causing TypeErrors (#31)

Version 0.2.5 (09 December 2013)

- Django 1.6 compatibility (#29)

Version 0.2.4 (30 May 2013)

- Fixed a packaging issue with 0.2.3

Version 0.2.3 (30 May 2013)

- Python3 compatibility!
- Code cleanup and code quality (flake8!)

- **[security]** Fixed billion laughs denial of service issue (#20)

Version 0.2.2 (22 May 2013)

- Fixed error involving simplejson with Django 1.5 (#14)
- Fixed ProtocolError relating to modwsgi and JSONEncoder (#15)
- Formatting cleanup

Version 0.2.1 (08 March 2013)

- Body length check fix for Django 1.5

Version 0.2.0 (04 November 2012)

- The demo site is now on Heroku
- `params` is now optional in JSONRPC requests (thanks to Albert Hopkins)
- Switching supported versions to Django supported versions (deprecating Django < 1.3 support)
- Moved documentation to readthedocs.org

Version 0.1.12 (02 February 2012)

- JSON encoding is customizable #3 (thanks to Alexander Morozov)

Version 0.1.11 (21 December 2011)

- Added a login_required parameter to the @rpcmethod decorator #2

Version 0.1.10 (14 November 2011)

- Removed requirement on Django in setup.py #798823
- Refactored to use standard setuptools tests (`python setup.py test`)
- Moved development to Github

Version 0.1.9 (10 July 2011)

- Added a CookieTransport class with a lot of help from Douglas Peter Sculley.
- RPC4Django's logging now goes to the rpc4django logger.
- Catches an ExpatError in xmlrpclib that was previously uncaught under certain conditions
- Fixed error with scanning of `INSTALLED_APPS` for `@rpcmethods`. This was causing an issue when South was installed. (#807628)
- Fixed bug #807653 related to scanning ServerProxy objects

Version 0.1.8 (26 October 2010)

- Added cross referenced Sphinx based documentation
- Fixed bug #570852 which caused incompatibilities with MongoDB because of the name class with the variable `is_rpcmethod`.
- Fixed bug #658788 which caused CSRF issues with `serve_rpc_request`.
- Added *out of the box authentication* as per the blueprint on Launchpad.

Version 0.1.7 (19 January 2010)

- Fixed a bug relating to CSRF
- Added feature to allow recursive imports of RPC methods

Version 0.1.6 (13 January 2010)

- Changed the XMLRPC dispatcher to allow sending nil which translates to the Python `None`. This was already allowed with JSONRPC
- Cross site request forgery (CSRF) framework support
- Added cross domain access control
- Added unit tests for base64 encoded binary
- Added access to HttpRequest object form inside a called RPC method

Version 0.1.5 (4 October 2009)

- Authenticated view that ties in with Django's auth system
- Added unicode unit test cases to verify that RPC4Django supports unicode (it does!)
- Added authenticated demo site (user = pass = rpc4django, self signed certificate)
- Improved the documentation stylesheet

Version 0.1.4 (31 August 2009)

- Provided a workaround for the bug relating to Django Bug #6681.
- Provided the settings.py option *`RPC4DJANGO_RESTRICT_REST`* which forces RPC4Django to not attempt to convert any of the method summary docstrings to restructured text.

Version 0.1.3 (15 July 2009)

- Fixed a serious bug where RPC4Django relied on `request.META['CONTENT_TYPE']` to be set by the web server
- Added generator tag to the template with RPC4Django and the version
- Renamed `RPC4DJANGO_RESTRICT_DOCUMENTATION` to *`RPC4DJANGO_RESTRICT_METHOD_SUMMARY`* for clarity
- Built out the example better so that it could be deployed as a demo without modification
- Improved JSON output formatting to be consistent with SimpleXMLRPCDispatcher's XML output

Version 0.1.2 (13 July 2009)

- Improved unit testing including adding support for testing views
- Fixed a minor bug involving the incorrect use of `xmlrpclib.Fault` which was causing some errors to be reported with the wrong error message
- Fetched the URL for the simple method descriptor using reverse()
- Enabled testing an RPC method directly from the method summary
- Fixed some failing unit tests on older python versions

Version 0.1.1 (11 July 2009)

- Improved documentation by integrating reST
- Allowed reST markup in rpcmethod docstrings
- Fix easy_install problems related to django templates
- Tested version compatibility
- Allowed analyzing post data when content-type cannot tell whether the data is XML or JSON

Version 0.1.0 (6 July 2009)

- First version of RPC4Django

Index

E

R