pyrs-resource Documentation

Release 0.2.0

Csaba Palankai

August 23, 2015

Contents

Project homepage: https://github.com/palankai/pyrs-resource

Documentation: http://pyrs-resource.readthedocs.org

Issue tracking: https://github.com/palankai/pyrs-resource/issues

CHAPTER 1

What is this package for

In the python world there are many RESTFul framework. Some of them based on Django others are based on Flask. I've tried some but I had the feeling, I want to learn one, the use with Django or Flask or even Odoo. And I don't mention sometimes I found them not flexible enough. So, I've decided write my own independent framework what anybody can use in at least the mentioned 3 different worlds.

CHAPTER 2

Nutshell (notice that, it would be the achivement)

```
from pyrs import resource
from pyrs.resource import GET

class UserResouce:

    @GET(response=ArrayOfUserSchema)
    def get_users(self):
        return User.objects.all()

    @PUT(path='/<int:user_id>', response=UserSchema, request=UserSchema)
    def update_user(self, user_id, body):
        user = get_object_or_404(User, pk=user_id)
        user.name = body['name']
        user.email = body['email']
        user.save()
        return user

app = resource.Application()
app.add('/user', UserResouce)
```

In this example I've shown Django (like) example. The schema is based on pyrs.schema. Even if I tend to use that framework, you would be able to use any other.

CHAPTER 3

Features

- Using simple classes or even functions (no inheritance)
- Wrapped error handling, errors can be serialised
- Extensible API
- Works with python 2.7, 3.3, 3.4 (tested against these versions)
- Hooks for extending the dispatching process

CHAPTER 4

Installation

```
$ pip install pyrs-resource
```

CHAPTER 5

Dependencies

See *requirements.txt* for details, but mainly depends on Werkzeug. I'm using that project routing capabilities. Also depends on *pyrs.schema* as I mentioned in nutshell section.

CHAPTER 6

Important caveats

This code right now really in beta state. I plan to release soon as possible a completely working code, but right now it's just shaping.

CHAPTER 7

The ecosystem

This work is part of pyrs framework. The complete framework follow the same intention to implement flexible solution.

CHAPTER 8

Contribution

I really welcome any comments! I would be happy if you fork my code or create pull requests. I've already really strong opinions what I want to achieve and how, though any help would be welcomed.

Feel free drop a message to me!

CHAPTER 9

Contents:

9.1 Application

class pyrs.resource.base.**App**(*hooks=None*, *resources=None*, ***config*)

Bases: object

Resource application, provide routing and execution

Parameters

- **hooks** (*list*) – List of hook classes (check *hooks*)
- **resources** (*list*) – Expected items *(path, resource class, [namespace])*
- **config** – optional configuration values (updated *conf*)

_add_class(*path*, *resource*, *prefix=''*)

_add_function(*path*, *resource*, *prefix=''*)

_make_rule(*path*, *methods*, *endpoint*)

add(*path*, *resource*, *prefix=''*)

add_rule(*rule*)

config = None

Store the configuration (copied from *conf*)

dispatch(*path_info*, *method*, *query=None*, *body=None*, *headers=None*, *cookies=None*, *session=None*)

handle_client_exceptions(*ex*, *path_info*, *method*, *opts=None*, *req=None*)

handle_exception(*ex*, *opts*, *req*)

hooks = []

resources = []

List of rules, will be **extended** by App(resources=[]) Tuple should be presented: ('path', Resource, [namespace])

set_function(*name*, *resource*)

setup_hooks()

transform_exception(*ex*)

9.2 Resource

pyrs.resource.resource.**DELETE**(*_func=None*, ***kwargs*)
Decorator function Ensure the given function will be available for GET method

pyrs.resource.resource.**GET**(*_func=None*, ***kwargs*)
Decorator function Ensure the given function will be available for GET method

pyrs.resource.resource.**PATCH**(*_func=None*, ***kwargs*)
Decorator function Ensure the given function will be available for GET method

pyrs.resource.resource.**POST**(*_func=None*, ***kwargs*)
Decorator function Ensure the given function will be available for POST method

pyrs.resource.resource.**PUT**(*_func=None*, ***kwargs*)
Decorator function Ensure the given function will be available for GET method

pyrs.resource.resource.**RPC**(*_func=None*, ***kwargs*)
Decorator function Ensure the given function will be available for POST method This action tend to use as Remote procedure call

pyrs.resource.resource.**endpoint**(*_func=None*, *path='/'*, ***kwargs*)
Deadly simple decorator, add options to the given function. Can be user with or without any keyword arguments. The default options would contain the path and the name of the function. Based on configuration: *conf.decorate*

9.3 Response handling

The entry point of the application wrapped by the Request. The request tend to be immutable.

Request actually is a builder, it builds request arguments for the endpoint, can hold extra information about the application about the whole environment and can be passed to the endpoint as well.

class pyrs.resource.request.**Request**(*opts*, *app=None*, *path=None*, *query=None*, *body=None*, *headers=None*, *auth=None*, *cookies=None*, *session=None*)
Bases: object

_get_inject(*name*, *force_kwargs=False*)

_inject(*inject*, *value*, *opt=None*)

_parse_value(*value*, *opt*)
Parse a value based on options. The option can be *None* means shouldn't be not parsed Can be an instance (or a subclass) of *schema.Object*. In that case the schema *load* will be executed

_setup_injects()

build()

9.4 Response handling

class pyrs.resource.response.**Response**(*content*, *app=None*, *opts=None*, *request=None*)
Bases: object

Generic response class

build()

setup()

9.5 Error handling

exception pyrs.resource.errors.**ClientError**(**args*, ***details*)
Bases: *pyrs.resource.errors.Error*

Generic Client Error. Normally the client errors have 4xx status codes.

status = 400

class pyrs.resource.errors.**DetailsSchema**(*extend=None*, ***attrs*)
Bases: pyrs.schema.types.Object

Details part of the error schema. Additional properties possible.

_attrs = OrderedDict([('additional', True)])

_definitions = None

_fields = OrderedDict([('traceback', <pyrs.schema.types.Array object at 0x7f42d0c8d4e0>), ('args', <pyrs.schema.type

exception pyrs.resource.errors.**Error**(**args*, ***details*)
Bases: Exception

This is the base exception of this framework. The response based on this exception will be a JSON data

description = None
Description of error. Should give details about the error In the message it will appearing as error_description

details = None
None used as empty dict. Gives extra information about this error which could be parsed by the consumer of API.

error = None
Error code should be a string. If it's not specified the class fully qualified name will be used

get_details(*debug=False*)
Gives back detailed information about the error and the context. By default its an empty dictionary. The *debug* depends on the debug parameter should give back traceback information and the positional arguments of the exception. As this is part of the message should conform with the *ErrorSchema*.

get_headers()
This method gives back the header property by default or an empty dict, but you can override, then provide special headers based on the context

get_message(*debug=False*)
Should give back a dictionary which will be threated the response body. The message should be conform with the *ErrorSchema*.

get_status()
This method gives back the status property by default which will be threated as HTTP status code. You can override, then provide your own status code based on the context.

headers = None
HTTP Response headers, (default None processed as empty)

schema = None
You can specify your schema class for validating your message By default the application default error schema the *ErrorSchema* will be used

status = 500
HTTP status code (default=500)

uri = None
Reference for this error. You can pointing out a documentation which gives more information about how could this error happen and how could be possible to avoid

classmethod wrap (*original*)
Wraps the exception gives back an *Error* instance. The created *Error* instance *error* property will be updated by the fully qualified name of the *original* exception. You could use it for *Error* instances as well, though is not recommended.

class pyrs.resource.errors.**ErrorResponse** (*content*, *app=None*, *opts=None*, *request=None*)
Bases: *pyrs.resource.response.Response*

setup ()

class pyrs.resource.errors.**ErrorSchema** (*extend=None*, ***attrs*)
Bases: pyrs.schema.types.Object

Describe how the error response should look like. Goal of this schema is a minimalistic but usable error response.

_attrs = OrderedDict([('additional', False)])

_definitions = None

_fields = OrderedDict([('error', <pyrs.schema.types.String object at 0x7f42d0c8d828>), ('error_description', <pyrs.sch

dump (*ex*)

exception pyrs.resource.errors.**InputValidationError** (**args*, ***details*)
Bases: *pyrs.resource.errors.Error*

error = 'invalid_request_format'

status = 400

exception pyrs.resource.errors.**ValidationError** (**args*, ***details*)
Bases: *pyrs.resource.errors.Error*

error = 'validation_error'

status = 500

9.6 Hooks

Hooks in general the way to override amend the exist functionality of app. Even you could extend the app, sometimes much easier if you attach a hook like authentication hook and the will process the request, make request.auth available. But also you can create your own hook handling special header values or give special error handling strategy.

The *Hook* class provide the skeleton of any further hooks.

class pyrs.resource.hooks.**Hook**
Bases: object

Hooks help to extend the functionality of application. The 3 hooks executed in different time of execution. This class should be the base class of any further hook.

exception (*request*, *exception*)
If the function raise any exception it can be handled with this hook. return will be used as response if it gives back any (should be *Response* instance or *None*)

request (*request*)
Executed when the request is created. It can amend the request. If has any return value it will be used as return value of the call, the the function will be not called. Can raise any exception and that will be treated as the function exception, in that case the function will be not called.

response (*response*)
Executed after successful call of the function. Response object created and passed to the hook. Can modify the response or give back a new response. Have to return the response object.

9.7 Configuration

This module contains the default configurations. The *pyrs.resource.base.App* config will be based on these values.

pyrs.resource.conf.**debug = False**
You can get more information in response like traceback and args of exception

pyrs.resource.conf.**decorate = '_endpoint'**
This option will be used for decorators. Usage *getattr(func, conf.decorate)*

pyrs.resource.conf.**host = 'localhost'**
Default host for the application

pyrs.resource.conf.**inject_app = False**
Enable/disable injecting the *base.App* as keyword argument

pyrs.resource.conf.**inject_app_name = 'app'**
Name used for app injection

pyrs.resource.conf.**inject_auth = False**
Enable/disable injecting the `request.auth` as keyword argument

pyrs.resource.conf.**inject_auth_name = 'auth'**
With this name the auth will be injected

pyrs.resource.conf.**inject_body = True**
Enable/disable injecting the request body

pyrs.resource.conf.**inject_cookies = False**
Enable/disable injecting the cookies

pyrs.resource.conf.**inject_cookies_name = 'cookies'**
With this name the cookies will be injected

pyrs.resource.conf.**inject_path = True**
Enable/disable injecting the path arguments If a name provided the path arguments will be injected as specified

pyrs.resource.conf.**inject_query = True**
Enable/disable injecting the query arguments If a name provided the query arguments will be injected as specified

CHAPTER 10

License

```
                   GNU LESSER GENERAL PUBLIC LICENSE
                       Version 3, 29 June 2007

 Copyright (C) 2007 Free Software Foundation, Inc. <http://fsf.org/>
 Everyone is permitted to copy and distribute verbatim copies
 of this license document, but changing it is not allowed.


  This version of the GNU Lesser General Public License incorporates
the terms and conditions of version 3 of the GNU General Public
License, supplemented by the additional permissions listed below.

  0. Additional Definitions.

  As used herein, "this License" refers to version 3 of the GNU Lesser
General Public License, and the "GNU GPL" refers to version 3 of the GNU
General Public License.

  "The Library" refers to a covered work governed by this License,
other than an Application or a Combined Work as defined below.

  An "Application" is any work that makes use of an interface provided
by the Library, but which is not otherwise based on the Library.
Defining a subclass of a class defined by the Library is deemed a mode
of using an interface provided by the Library.

  A "Combined Work" is a work produced by combining or linking an
Application with the Library.  The particular version of the Library
with which the Combined Work was made is also called the "Linked
Version".

  The "Minimal Corresponding Source" for a Combined Work means the
Corresponding Source for the Combined Work, excluding any source code
for portions of the Combined Work that, considered in isolation, are
based on the Application, and not on the Linked Version.

  The "Corresponding Application Code" for a Combined Work means the
object code and/or source code for the Application, including any data
and utility programs needed for reproducing the Combined Work from the
Application, but excluding the System Libraries of the Combined Work.

  1. Exception to Section 3 of the GNU GPL.
```

```
  You may convey a covered work under sections 3 and 4 of this License
without being bound by section 3 of the GNU GPL.

  2. Conveying Modified Versions.

  If you modify a copy of the Library, and, in your modifications, a
facility refers to a function or data to be supplied by an Application
that uses the facility (other than as an argument passed when the
facility is invoked), then you may convey a copy of the modified
version:

   a) under this License, provided that you make a good faith effort to
   ensure that, in the event an Application does not supply the
   function or data, the facility still operates, and performs
   whatever part of its purpose remains meaningful, or

   b) under the GNU GPL, with none of the additional permissions of
   this License applicable to that copy.

  3. Object Code Incorporating Material from Library Header Files.

  The object code form of an Application may incorporate material from
a header file that is part of the Library.  You may convey such object
code under terms of your choice, provided that, if the incorporated
material is not limited to numerical parameters, data structure
layouts and accessors, or small macros, inline functions and templates
(ten or fewer lines in length), you do both of the following:

   a) Give prominent notice with each copy of the object code that the
   Library is used in it and that the Library and its use are
   covered by this License.

   b) Accompany the object code with a copy of the GNU GPL and this license
   document.

  4. Combined Works.

  You may convey a Combined Work under terms of your choice that,
taken together, effectively do not restrict modification of the
portions of the Library contained in the Combined Work and reverse
engineering for debugging such modifications, if you also do each of
the following:

   a) Give prominent notice with each copy of the Combined Work that
   the Library is used in it and that the Library and its use are
   covered by this License.

   b) Accompany the Combined Work with a copy of the GNU GPL and this license
   document.

   c) For a Combined Work that displays copyright notices during
   execution, include the copyright notice for the Library among
   these notices, as well as a reference directing the user to the
   copies of the GNU GPL and this license document.

   d) Do one of the following:

       0) Convey the Minimal Corresponding Source under the terms of this
```

```
       License, and the Corresponding Application Code in a form
       suitable for, and under terms that permit, the user to
       recombine or relink the Application with a modified version of
       the Linked Version to produce a modified Combined Work, in the
       manner specified by section 6 of the GNU GPL for conveying
       Corresponding Source.

       1) Use a suitable shared library mechanism for linking with the
       Library.  A suitable mechanism is one that (a) uses at run time
       a copy of the Library already present on the user's computer
       system, and (b) will operate properly with a modified version
       of the Library that is interface-compatible with the Linked
       Version.

   e) Provide Installation Information, but only if you would otherwise
   be required to provide such information under section 6 of the
   GNU GPL, and only to the extent that such information is
   necessary to install and execute a modified version of the
   Combined Work produced by recombining or relinking the
   Application with a modified version of the Linked Version. (If
   you use option 4d0, the Installation Information must accompany
   the Minimal Corresponding Source and Corresponding Application
   Code. If you use option 4d1, you must provide the Installation
   Information in the manner specified by section 6 of the GNU GPL
   for conveying Corresponding Source.)

  5. Combined Libraries.

  You may place library facilities that are a work based on the
Library side by side in a single library together with other library
facilities that are not Applications and are not covered by this
License, and convey such a combined library under terms of your
choice, if you do both of the following:

   a) Accompany the combined library with a copy of the same work based
   on the Library, uncombined with any other library facilities,
   conveyed under the terms of this License.

   b) Give prominent notice with the combined library that part of it
   is a work based on the Library, and explaining where to find the
   accompanying uncombined form of the same work.

  6. Revised Versions of the GNU Lesser General Public License.

  The Free Software Foundation may publish revised and/or new versions
of the GNU Lesser General Public License from time to time. Such new
versions will be similar in spirit to the present version, but may
differ in detail to address new problems or concerns.

  Each version is given a distinguishing version number. If the
Library as you received it specifies that a certain numbered version
of the GNU Lesser General Public License "or any later version"
applies to it, you have the option of following the terms and
conditions either of that published version or of any later version
published by the Free Software Foundation. If the Library as you
received it does not specify a version number of the GNU Lesser
General Public License, you may choose any version of the GNU Lesser
General Public License ever published by the Free Software Foundation.
```

```
  If the Library as you received it specifies that a proxy can decide
whether future versions of the GNU Lesser General Public License shall
apply, that proxy's public statement of acceptance of any version is
permanent authorization for you to choose that version for the
Library.
```

10.1 Indices and tables

- genindex
- modindex
- search

p

Symbols

A

B

C

D

E

G

H

I

P

R

S

T

U

V

W