link.wsgi Documentation

Release 0.3

David Delassus

May 25, 2016

Contents

link.wsgi is a fully configurable WSGI microframework. The only needed developments are routes and eventually middlewares.

The rest is done via the configuration of the library, allowing the reuse of the code.

Check out the source code on Github.

Contents:

CHAPTER 1

Tutorial

In this tutorial, we will build a new package containing routes for our WSGI applications and then deploy everything in a virtualenv.

1.1 Simple route

A simple route is just a function which takes two parameters:

- the received request
- the response to fill

For example:

```
def hello(req, resp):
    resp.status = 200
    resp.content = 'Hello world'
```

See the API of *Request* and *Response* objects for more informations.

1.2 Class-based view

A route is just a callable object, if a class is used, then it will be instantiated. The called handler is a method named after HTTP verbs:

```
class Hello(object):
    def get(self, req, resp):
        resp.status = 200
        resp.content = 'Hello world'
```

1.3 A simple middleware

Middlewares are objects which are applied to the request before and after the handling of the request.

It can be any objects respecting the API of the *Middleware* class.

Example:

```
from link.wsgi.middleware import Middleware


class MyMiddleware(Middleware):
   def before(self, req, resp, handler):
      # do something with request
      # do something with resp
      # do something with handler
      return False  # True to abort the request

   def after(self, req, resp, handler):
      # do something with request
      # do something with resp
      # do something with handler
```

1.4 Routes configuration

Configuration file for the router is stored in:

```
$B3J0F_CONF_DIR/link/wsgi/router.conf
```

Here is an example of configuration where the simple route `hello()` is in the package `myapp.routes` and the middleware `MyMiddleware` is in the package `myapp.middlewares`:

```
{
   "ROUTER": {
      "urlpatterns": {
         "^/hello$": {
            "GET": "mapp.routes.hello"
         }
      },
      "middlewares": {
         "myapp.middlewares.MyMiddleware"
      }
   }
}
```

1.5 Deploying

1.5.1 Prerequisites

Make sure the command `virtualenv` is available.

1.5.2 Creating virtualenv

Assuming you're in your Python package folder:

```
$ virtualenv myapp-venv
$ . ./myapp-venv/bin/activate
(myapp-venv)$ pip install supervisord gunicorn link.wsgi
(myapp-venv)$ python setup.py install
```

1.5.3 Configuring the whole thing

We need the following *supervisord* service:

```
[program:myapp]

environment=B3J0F_CONF_DIR="%(ENV_VIRTUAL_ENV)s/etc"
command=gunicorn link.wsgi.app:application
```

1.6 Running everything

```
(myapp-venv)$ supervisord
(myapp-venv)$ supervisorctl start myapp
(myapp-venv)$ curl http://localhost:8000/hello
Hello world
```

CHAPTER 2

API documentation

2.1 link.wsgi package

2.1.1 Submodules

2.1.2 link.wsgi.app module

class link.wsgi.app.**Application**(**args*, ***kwargs*)
Bases: object

WSGI Application class.

2.1.3 link.wsgi.middleware module

class link.wsgi.middleware.**Middleware**(**args*, ***kwargs*)
Bases: object

Middleware class.

Applied before and after requests are handled.

after(*req*, *resp*, *handler*)
Called after request is handled.

Parameters

- **req** (link.wsgi.req.Request) – request that was handled
- **resp** (link.wsgi.resp.Response) – response that was returned
- **handler** (*callable*) – handler that was used

before(*req*, *resp*, *handler*)
Called before request is handled.

Parameters

- **req** (link.wsgi.req.Request) – request that will be handled
- **resp** (link.wsgi.resp.Response) – response that will be returned
- **handler** (*callable*) – handler that will be used

Returns True to abort request handling

Return type boolean

2.1.4 link.wsgi.req module

class `link.wsgi.req.`**Request**(*environ*, **args*, ***kwargs*)
Bases: `object`

Request object encapsulating WSGI environ dict.

charsets

content

content_length

content_type

method

path

query

2.1.5 link.wsgi.resp module

class `link.wsgi.resp.`**Response**(*start_response*, **args*, ***kwargs*)
Bases: `object`

Response object encapsulating WSGI response handler.

content

headers

status

2.1.6 link.wsgi.router module

class `link.wsgi.router.`**Router**(*urlpatterns=None*, *middlewares=None*, **args*, ***kwargs*)
Bases: `object`

Request dispatcher.

Contains URL patterns as dict:

- a regex to match the URL as key
- a dict associated HTTP methods to Python callable objects

Also contains list of middlewares (Python classes) to apply.

Example of configuration:

```
{
   "ROUTER": {
      "urlpatterns": {
         "^/hello$": {
            "GET": "python.path.to.hello_function"
         }
      },
      "middlewares": [
         "python.path.to.MiddlewareClass"
      ]
```

```
        }
    }
```

dispatch(*req*, *resp*)
Dispatch request to handler, which will fill response.

Parameters

- **req** (link.wsgi.req.Request) – request object
- **resp** (link.wsgi.resp.Response) – response object

middlewares

urlpatterns

2.1.7 link.wsgi.url module

link.wsgi.url.**parse_qs**(*query*)
Override six.moves.urllib.parse.parse_qs to handle array parameters

2.1.8 Module contents

l