Jokk Documentation

Release 0.1

Shinya Ohyanagi

January 05, 2014

Contents

CHAPTER 1

What is Jokk?

RESTful mock api server.

Jokk can provide HTTP Response mock data easly.

```
GET  /user/1 => user/1_get.json
POST /user/1 => user/1_post.json
```

Jokk is heavily inspiered by EasyMock.

Naming of *Jokk* is inspiered by JokkMokk because pronunciation is similar to *Mock*.

- Repository
- Documentation

CHAPTER 2

Installation

```
$ virtualenv --distribute jokk_sample
$ source jokk_sample/bin/activate
$ cd jokk_sample
$ pip install jokk
```

Jokk depends on Werkzeug using for WSGI Utility Library.

CHAPTER 3

Usage

1. Create *config.json* for configure settings such as routes, variables
2. Create *data* directory for serve response files
3. Put response file into *data* directory
4. Start Jokk server

```
$ jokk -c config.json
```

5. Access to jokk server client such as Web browser.
6. Jokk would return response file

CHAPTER 4

Configure settings

Configure settings like below.

```
{
  "data": "./data",
  "jsonp": true,
  "cors": true,
  "routes": [
    "/user",
    "/user/<userid>"
    "/user/<userid>/show"
  ],
  "variables" : {
    "server": "http://example.com"
  }
}
```

Key	Value
data	Path to data directory for serve response file
jsonp	Enable to use JSONP
cors	Enable to use Cross-Origin Resource Sharing
routes	Routes to serve response file
variables	Enable to assign setting key-value to response body

4.1 data

Defined relative path from *config.json*.

For example, if you defined *./data* in *config.json*, directory structors should be like following.

```
-config.json
-data
   -user_get.json
   -user
     -userid_get.json
     -userid_post.json
```

4.2 JSONP and CORS

jsonp and *cors(Cross-Origin Resource Sharing)* are for cross domain access.

- When *jsonp* value is true, response body would add callback method.
- When *cors* value is true, response header would add following header.

Header name	Header value
Access-Control-Allow-Origin	*
Access-Control-Allow-Methods	GET,PUT,POST,DELETE,PATCH
Access-Control-Allow-Headers	Content-Type, Authorization

4.3 routes

Routes for serve response file. When url rules matched, Jokk will search response file and status file.

Rules	Response file path
/	./data/_get.json
/user	./data/user_get.json
/user/<userid>	./data/userid_get.json
/user/<userid>/<id>	./data/userid/id_get.json

Convention of file name is following.

```
Url rules + '_' + HTTP method + {.json,.xml,.html,txt}
Url rules + '_' + HTTP method + .status
```

HTTP method	rutes	response file name
GET	/items/1	items/1_get.json
POST	/items/1	items/1_post.json
PUT	/items/1	items/1_put.json
DELETE	/items/1	items/1_delete.json
PATCH	/items/1	items/1_patch.json
HEAD	/items/1	[1]

4.3.1 Status file

If you want to send custom status code, put status file such as *item/1_get.stasus* into same directory as response file.

In status file you just put integer value like following.

```
201
```

4.3.2 Response file types

Following response file types are available.

File type	MimeType
json	application/json
xml	application/xml
html	text/html
text	text/plain

[1] HEAD method returns empty response body.

4.3.3 Variable Rules

To add variable parts to a URL you can mark these special sections as <variable_name> and the given name will be available as a variable.

For example, routes defined such as */user/<userid>* in *config.json*, and call *GET /user/1234* would be mapped to *./data/user/userid_get.json*.

You can write variable in response file.

```
{
  "userid": "${userid}"
}
```

Above response would be replaced to following.

```
{
  "userid": "1234"
}
```

4.4 variables

If you define *variables* in *config.json*, you can use in response file.

```
{
  "variables" : {
    "server": "http://example.com"
  }
}
```

Define response file like following.

```
{
  "server": "${server}"
}
```

Response body would be replaced like following.

```
{
  "server": "http://example.com"
}
```

CHAPTER 5

API

class jokk.server.**Jokk**(*config_path*)

create_response(*request*, *params*, *endpoint*)
Read response file(json,xml,html,txt), status file.

HTTP request	JSON file.
GET /user/1	./data/user/1_get.json
POST /user	./user/post.json
GET /user/<id>	./user/id_get.json
HEAD /user/1	empty string

Find response file json, xml, html, txt order.

Parameters

- **request** – Request object.
- **params** –
- **endpoint** – Base response file name rule.

create_url_map(*routes*)
Create url_map.

Parameters **routes** – Routing dict

data_path = None
Served data path's root is same as *config.json* path.

```
-config.json
-data
  -user_get.json
```

dispatch(*request*)
Dispatch HTTP requests and create response.

Parameters **request** – Werkzeug request

read_config(*path*)
Read config.json and load to dict.

Parameters **path** – Path to config.json

url_map = None
Routings are defined in config.json.

wsgi_app(*environ*, *start_response*)
Create WSGI response.

Parameters

- **environ** – Environmen
- **start_response** – Response

class jokk.server.**parse_option**
Parse options.

Options	Default	Description
-b, –bind	127.0.0.1	Mock server url
-p, –port	5000	Port number
-d, –debug	True	Show tracelog
-r, –reloader	False	Auto reloader
-s, –show_urls	False	Show urls
-c, –config	None	Config file

class jokk.server.**show_urls**
Displays all of the url matching routes.

Parameters **app** – Jokk object

CHAPTER 6

ChangeLog

CHAPTER 7

Version 0.1

Released on January 20th 2013

- First public preview release.

Contributing

1. Fork it
2. Create your feature branch (git checkout -b my-new-feature)
3. Commit your changes (git commit -am ‘Add some feature’)
4. Push to the branch (git push origin my-new-feature)
5. Create new Pull Request

CHAPTER 9

Indices and tables

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