apicore Documentation

Release 1.0

dev@meez.io

Feb 09, 2018

Contents

Set of core libraries for building REST API and Microservices based on Flask.

The code is open source, release under MIT and written in Python 3.

```
pip install apicore
```

CHAPTER 1

Features

- Cross-origin resource sharing (CORS) ready
- Data caching with redis server or direct in-memory
- Configuration file loader
- A simple Logger
- Raise exception conform to HTTP status codes
- OpenAPI 3.0 specification embedded with Swagger UI

CHAPTER 2

Example

```
#!/usr/bin/env python

from apicore import api, Logger, config, Http409Exception

Logger.info("Starting {} API Server...".format(config.app_name))


@api.route('/error/')
def error():
    """
    summary: Raise an execption
    responses:
      409:
          description: Conflict
    """
    raise Http409Exception()


if __name__ == "__main__":
    # api is an instance of API which inherit from Flask
    api.debug = config.debug
    api.run()
```

CHAPTER 3

Configuration

Configuration is set in `conf/config.yaml` file (see `apicore.config.Config`).

Name	Default value	Description
app_name	"My App"	Application's name.
debug	True	Active debug mode.
prefix	""	Add a prefix to all URL paths (ie: "/api").
redis	None	Redis server used for caching data : redis://:password@localhost:6379/0. If not set use in-memory.
swagger_ui	"/"	Relative URL path to embedded Swagger UI (`prefix` + `swagger_ui`).
specs_login	None	Login to access API Specification (`openapi.json`), no Authentification by default.
specs_pwd	None	Password to access API Specification.

CHAPTER 4

OpenAPI 3.0

- See specification for syntax.
- Document route's methods with Operation Object using yaml syntax.
- Document your API in `conf/openapi.yaml` file.
- Access your documentation through a python dictionary : `api.oas.specs`.
- Your spec is available at `http[s]://<hostname>/openapi.json`.
- Default path to `http[s]://<hostname>/` to see your spec with Swagger UI (set `swagger_ui` in `conf/config.yaml` to change path)
- Full exemple :

```
@api.route('/sellers/<idseller>/', methods=['GET', 'PUT'])
def seller(idseller):
    """
    description: "Path Item Object" level here, only common_responses is added to␣
↪OpenAPi specification. Next level are "Operation Object".
    parameters:
      - name: idseller
        in: path
        description: uuid of seller
        required: true
        type: string
        format: uuid
    common_responses:
        400:
         description: Invalid request
        401:
          description: Authentification required
        403:
          description: Ressource access denied
        500:
          description: Server internal error
    ---
```

```
    tags:
      - profile
    summary: Find a seller profile by ID
    responses:
      200:
        description: Success
        content:
          application/json:
            schema:
              $ref: '#/components/schemas/Seller'
      404:
        description: Ressource does not exist
      406:
        description: Nothing to send maching Access-* headers
  ---
    tags:
      - profile
    summary: Update seller profile
    requestBody:
      content:
        application/json:
          schema:
            $ref: '#/components/schemas/Seller'
        required: true
    responses:
      200:
        description: Success
  """
  pass


  print(api.oas.spec)
```

CHAPTER 5

APIs

5.1 api

`api` is the application, instance of `apicore.api.API` inherited from `flask.Flask`. It handle Cross-origin resource sharing (CORS) and JSON responde message (instead of HTML).

5.2 cache

`cache` is an instance of `apicore.cache.Cache`

5.3 config

`config` is an instance of `apicore.config.Config`

5.4 Exceptions

5.5 Lang

5.6 Logger

Todo:

- i18n HTTP response messages.
- Add namespace for cache

- Configure using command line argument and environnement variables which override configuration file and making it optional.
- Use API Specification and json schemas to validate JSON data
- Access Control Policies engine
- MongoDB helpers
- Extensible notification system (using mail, Firebase, SMS, . . .)