Mockerena Documentation

Release 1.3.1

Michael Holtzscher

Dec 11, 2019

CONTENTS:

Mockerena allows anyone to mock data files through customized json schemas and provides routes for external systems to retrieve from it.

CHAPTER

ONE

GETTING STARTED

Install through git:

```
git clone https://github.com/FanThreeSixty/mockerena
```

Either you can install the project yourself or you can use the prepared scripts

To use the prepared scripts use:

```
cd mockerena
script/setup
```

To install dependencies yourself:

```
cd mockerena

# For Unix environments
virtualenv -p python3 venv
source venv/bin/activate

# For Windows environments
python -m venv venv/
venv/Scripts/activate.bat

pip install -r requirements.txt
pip install -e .
```

To run the project either use:

```
script/server
```

Or manually through:

```
python mockerena/app.py
```

CHAPTER

TWO

DEVELOPMENT

2.1 Generating documentation

You can either use makefile:

```
cd docs
make html
```

Or you can use autobuild:

```
cd docs
sphinx-autobuild . _build/html/
```

Or you can use the script:

```
script/docs
```

2.2 Linting project

You can run pylint yourself:

```
pylint mockerena
```

Or you can use the script:

```
script/lint
```

2.3 Running tests

Testing requires 3 python libraries to be installed prior to running. If you use the `test` script you can avoid installing these dependencies yourself. To install dependencies run:

```
pip install pytest pytest-cov pytest-flask
```

Once the dependencies are installed you can run pytest yourself:

```
pytest tests/
```

Or you can use the script:

```
script/test
```

2.4 Updating dependencies

You can run pip update yourself:

```
pip install -U -r requirements.txt
```

Or you can use the script:

```
script/update
```

CHAPTER

THREE

DEPLOYMENT

3.1 Deploy to AWS

First you'll need to install the AWS CLI and configure. For more information on these steps, please visit Amazon's documentation for further details. Once the CLI is installed you'll need to install the serverless cli:

```
npm install -g serverless
```

Afterwards, to deploy simply run:

```
serverless deploy
```

Once the deploy is complete, run `sls info` to get the endpoint:

```
sls info
```

3.2 Docker deployment

```
docker build -t mockerena .
docker run -it -p 5000:5000 mockerena

# Or using docker compose
docker-compose up -d
```

3.3 Local deployment

To run a serverless setup locally, run:

```
sls wsgi serve
```

Navigate to localhost:5000 to see your app running locally.

Or you can deploy manually running this command in your `init.d` or `systemctl` scripts:

```
gunicorn3 --config gunicorn_config.py mockerena.app:app
```

3.4 Configuration

To configure Mockerena, inside `settings.py` there are a few settings:

```
HOST = os.environ.get('MOCKERENA_HOST', 'localhost')
PORT = os.environ.get('MOCKERENA_PORT', 9000)
BASE_PATH = os.environ.get('MOCKERENA_BASE_PATH', '')
DEBUG = os.environ.get('MOCKERENA_DEBUG', False)
SECRET_KEY = os.environ.get('MOCKERENA_SECRET_KEY', None)
ENV = os.environ.get('MOCKERENA_ENV', 'development')
```

At a minimum at least update `MOCKERENA_HOST` and `MOCKERENA_PORT` to whatever the new host and port will be and `MOCKERENA_SECRET_KEY` to a random hash.

There are also settings for configuring Mongo with mockerena. Update these as necessary:

```
# Database settings
MONGO_HOST = os.environ.get('MOCKERENA_MONGO_HOST', 'localhost')
MONGO_PORT = os.environ.get('MOCKERENA_MONGO_PORT', 27017)
MONGO_DBNAME = os.environ.get('MOCKERENA_MONGO_DBNAME', 'mockerena')
MONGO_AUTH_SOURCE = os.environ.get('MOCKERENA_MONGO_AUTH_SOURCE', 'mockerena')
MONGO_USERNAME = os.environ.get('MOCKERENA_MONGO_USERNAME', '')
MONGO_PASSWORD = os.environ.get('MOCKERENA_MONGO_PASSWORD', '')
```

For more configuration options visit Eve's documentation

CHAPTER

FOUR

USING MOCKERENA

To generate fake data you can either:

1) Save schemas and generate data using an id or
2) You can generate data on-the-fly.

To save a schema, POST to `/api/schema`:

```
{
    "schema": "mock_example",
    "num_rows": 10,
    "file_format": "csv",
    "file_name": "mock_{}_example",
    "include_header": true,
    "delimiter": ",",
    "quote_character": "'",
    "columns": [
        {
            "name": "foo",
            "truncate": false,
            "type": "word",
            "description": "First column",
            "percent_empty": 0.2
        },
        {
            "name": "bar",
            "type": "random_element",
            "description": "Second column",
            "truncate": false,
            "args": {
                "elements": ["that"]
            },
            "function": "this + this"
        }
    ]
}
```

To breakdown what is happening at the schema-level above, here are what each item means:

schema - The name of the schema

num_rows - The default number of records to return

file_format - The default format of the file

file_name - The name of the file that is generated. *{}* is used to insert a datetime

include_header - Include the header for the CSV

exclude_null - Squash nulls for JSON formats

is_nested - Generate nested JSON

delimiter - CSV column separator

quote_character - Quoting character for CSV or TSV

On a column-level:

name - Column header name

type - Faker data type. See `/api/types` for a list of all types

description - A description of what what the column is for

truncate - Drop column after generation

percent_empty - Likeliness that the column will be empty. 0 to 1, 1 being 100%

args - Arguments passed into type

function - Post-processing function (*see below*)

format - Date format (use standard python date format strings)

To generate data, GET to `/api/schema/{schema_id}/generate`. You should receive something like this:

```
foo,bar
lose,thatthat
now,thatthat
and,thatthat
,thatthat
such,thatthat
government,thatthat
around,thatthat
room,thatthat
behind,thatthat
television,thatthat
```

You can optionally POST to `/api/schema/generate` directly to generate data without having to permanently save the schema.

4.1 Functions

Mockerena mostly uses `Faker` providers to generate random data. Click here for the full list of providers from `Faker`. With Mockerena, we've supplied a few additional providers that are available here.

You can also use the types endpoint `/api/types` to retrieve a complete list of all provider types.

4.2 Templates

Mockerena gives users the flexibility to define return types that aren't listed as options for file_format. Any file format that isn't immediately recognized, requires a template. So for example:

```
{
    "schema": "mock_example",
    "num_rows": 10,
```

(continues on next page)

(continued from previous page)

```
    "file_format": "xml",
    "file_name": "mock_{}_example",
    "template": "<root>{% for r in records %}<record><foo>{{r['foo']}}</foo><bar>{{r[
↪'bar']}}</bar></record>{% endfor %}</root>"
    "columns": [
        {
            "name": "foo",
            "truncate": false,
            "type": "word",
            "description": "First column",
            "percent_empty": 0.2
        },
        {
            "name": "bar",
            "type": "random_element",
            "description": "Second column",
            "truncate": false,
            "args": {
                "elements": ["that"]
            },
            "function": "this + this"
        }
    ]
}
```

Would return and XML response like:

```
<root>
    <record>
        <foo>lose</foo>
        <bar>thatthat</bar>
    </record>
    <record>
        <foo>now</foo>
        <bar>thatthat</bar>
    </record>
    <record>
        <foo>and</foo>
        <bar>thatthat</bar>
    </record>
    <record>
        <foo></foo>
        <bar>thatthat</bar>
    </record>
    ....
</root>
```

And since Mockerena uses Jinja2 as the templating engine, you can leverage their robust set of filters and tests to further control how data populates the template.

4.3 Responses

Added v1.1.0

Responses allow for custom responses to be randomly returned For example:

```
{
    "schema": "mock_example",
    "file_format": "csv",
    "file_name": "mock_{}_example",
    "columns": [
        {
            "name": "foo",
            "type": "word"
        }
    ],
    "responses": [
        {
            "status_code": 201,
            "weight": 2
        },
        {
            "status_code": 502,
            "data": "",
            "content_type": "text/plain",
            "headers": {
                "Last-Modified": "Thur, 19 Sep 2019 19:25:10 GMT"
            },
            "weight": 1
        }
    ]
}
```

To breakdown what is happening:

status_code - Override response code returned. Default is 200

data - Override data returned. Default is usual dataset

content_type - Override content type. Default is based off *file_format*, or "*text/plain*"

headers - Override response headers

weight - Probability response is returned. For example, a response with a weight of 2 is twice as likely to return than a response with a weight of 1

So in the example approximately 2 out of 3 attempts will return the normal response with a status code of *201*, but 1 out of 3 attempts will return a response with a status code of *502*, empty content and a last modified header with the timestamp "*Thur, 19 Sep 2019 19:25:10 GMT*".

CHAPTER

FIVE

API ROUTES

5.1 Retrieve all schemas

GET `/api/schema`

Retrieves one or more schema

Query params (optional):

Parameter	Description
projection	Conditional query where the user dictates which fields should be returned by the API

5.2 Store a schema

POST `/api/schema`

Stores one or more schema

5.3 Retrieve a schema

GET `/api/schema/{schema_id}`

Retrieves a schema document

Parameter	Description
schema_id	Either the name or the id of the schema

5.4 Replace a schema

PUT `/api/schema/{schema_id}`

Replaces a schema document

Parameter	Description
schema_id	Either the name or the id of the schema

5.5 Update a schema

PATCH `/api/schema/{schema_id}`

Updates a schema document

Parameter	Description
schema_id	Either the name or the id of the schema

5.6 Delete a schema

DELETE `/api/schema/{schema_id}`

Deletes a schema document

Parameter	Description
schema_id	Either the name or the id of the schema

5.7 Generate data

GET `/api/schema/generate`

Generates sample data for a provided schema

Query params (optional):

Parameter	Description
seed	The seed to use for the data generator
numrows	The number or rows of data to generate
file_format	Format of output
include_header	Include header with CSV, TSV or template
exclude_null	Squash nulls for JSON output

5.8 Generate data

GET `/api/schema/{schema_id}/generate`

Generates sample data from a schema

Parameter	Description
schema_id	Either the name or the id of the schema

Query params (optional):

Parameter	Description
seed	The seed to use for the data generator
numrows	The number or rows of data to generate
file_format	Format of output
include_header	Include header with CSV, TSV or template
exclude_null	Squash nulls for JSON output

5.9 Get provider types

DELETE `/api/types`

Returns all available provider types

CHAPTER

SIX

API REFERENCE

6.1 Mockerena

6.1.1 app

6.1.2 providers

Provider types not covered by vanilla Faker

class mockerena.providers.**MockProvider**(*generator*)

Bases: faker.providers.BaseProvider

Provider instance for types not supported by Faker

empty() → str

Returns an empty response

Returns An empty string

Return type str

price(*minimum: int = 0*, *maximum: int = 999999*) → float

Returns a random price within the range provided

Parameters

- **minimum** (*int*) – Minimum price
- **maximum** (*int*) – Maximum price

Returns Random price within range

Return type float

regex(*expression: str = ''*) → str

Returns a string generated from a regular expression

Parameters **expression** (*str*) – Regular expression

Returns A string generated from a regular expression

Return type str

weighted_choice(*elements: list*, *weights: list = None*) → Any

Returns a random element from a list of weighted choices

Parameters

- **elements** (*list*) – List of choices

- **weights** (*list*) – Weights to give each choice. Must be equal length to elements

Returns A single element from the list

Return type Any

Raises AssertionError

CHAPTER

SEVEN

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

m

INDEX