Published in Quantrium.ai

Bhargav Sridhar Follow
Jul 30, 2021 · 5 min read · Listen

Save

QUANTRIUM GUIDES

Creating a Hello World API using Swagger UI and Python

Getting started with your first Swagger UI document

Swagger is a web-based API documentation framework. It is used to create interactive

- Enable you to create and share API
- Allows you to test APIs

In this article, I explain step-by-step pro
a "Hello World" response through an AP
will use Python and YAML files to imple

As a pre-requisite, you are expected to h
how they work. If you need to gain a basic understanding on Flask API's, you can refer to the official Flask RESTful API documentation.

Steps to Create a Swagger UI Document

We will follow the following steps to build a Swagger UI document for an API function:

- First, we will create the API using Flask web API framework.
- Next, we will create a JSON or a YAML file to implement API functionality in SwaggerUI.
- Finally, we will call the created JSON or YAML file inside the Python program where the API definition is present.

You need to install the following Python packages using `pip install` to run this example.

```
$ pip install Flask
$ pip install flasgger
```

If you want to update the module while installing it, you can use `pip install -U <module_name>` or if you are using Python virtual environments and wish to install a specific version of the module for your project, you can execute `pip install`

```
from flask import Flask, request
from flasgger import Swagger, La
from flasgger import swag_from
```

Define the Flask app using the Flask me

```
app = Flask(__name__)
```

You need the JSON encoder to create JSON from the API objects for which you can use LazyJSONEncoder class.

```
app.json_encoder = LazyJSONEncoder
```

The template and the configuration for the Swagger UI document are defined as dictionary objects as follows:

```
swagger_template = dict(
info = {
    'title': LazyString(lambda: 'My first Swagger UI document'),
    'version': LazyString(lambda: '0.1'),
    'description': LazyString(lambda: 'This document depicts a
sample Swagger UI document and implements Hello World functionality
after executing GET.'),
    },
    host = LazyString(lambda: request.host)
```

```
            {
                "endpoint": 'hello_w
                "route": '/hello_wor
                "rule_filter": lambd
                "model_filter": lamb
            }
        ],
        "static_url_path": "/flasgger_static",
        "swagger_ui": True,
        "specs_route": "/apidocs/"
    }
```

The `LazyString` functionality of `flasgger` module is used to set default values for certain parameters during runtime. The parameters of the Swagger UI document such as document title, version, description etc. are defined within the `info` field. The base URL for the API is specified within the `host` field. The configuration details of Swagger UI are defined within the `swagger_config` JSON object.

Finally, the swagger object is defined using the Swagger method imported from `flasgger` module as follows:

```
swagger = Swagger(app, template=swagger_template,
                  config=swagger_config)
```

The API function is defined and the route/address of the Flask app is specified using the `@app.route` decorator. The YAML file for Swagger UI document is also called using the `@swag_from` decorator by specifying path of YAML file and `GET` method as arguments (arguments such as `POST`, `PUT`, `DELETE` can also be used as per

```
@app.route("/")
def hello_world():
    return "Hello World!!!"
```

> **NOTE:** *In this case, the YAML file and the*
> *the correct file path of the YAML file shou*
> *argument.*

The following lines should be added inside the `hello_world.yml` file for implementing the `GET` response in Swagger UI:-

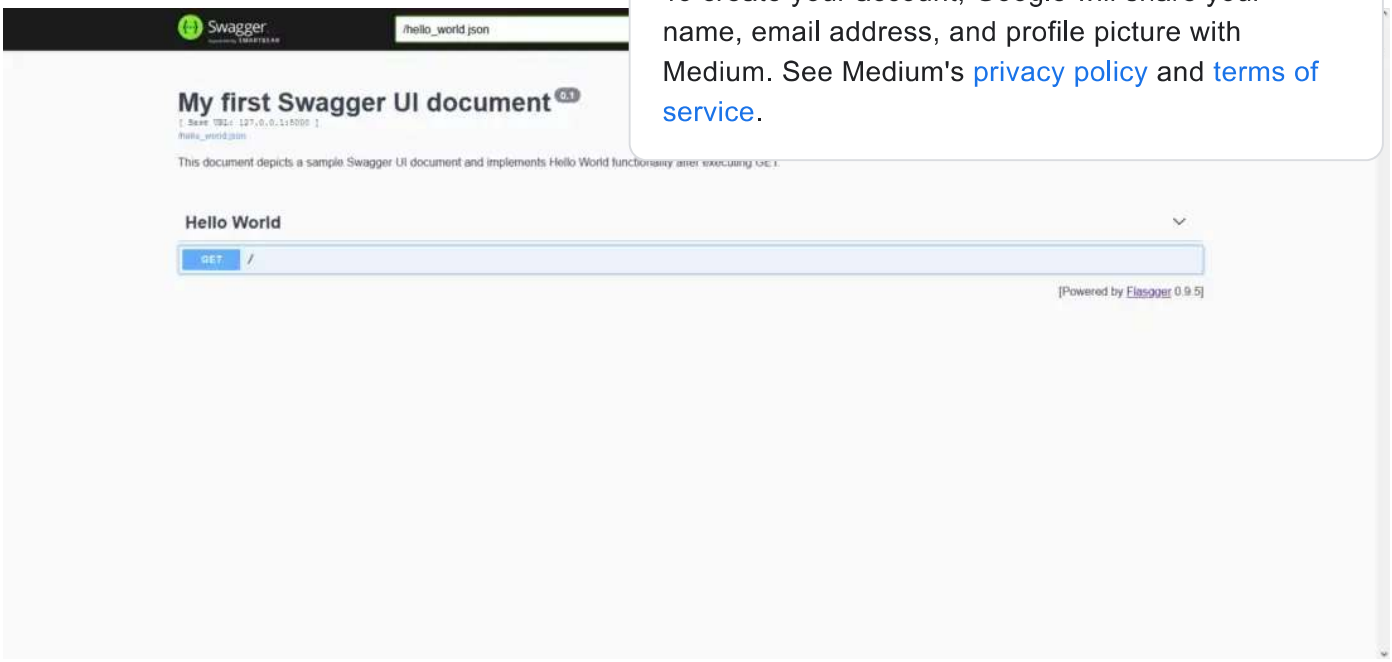
```
openapi: 3.0.0
tags:
  - name: Hello World
get:
  description: None
responses:
    '200':
      description: Successful response
    '400':
      description: Bad Request
    '500':
      description: Internal Server Error
```

> **NOTE:** *If the GET response does not have any parameters to be filled, it is a good practice to specify description as None and define responses in the end.*

Finally, the Flask app is run and executed inside the main using the following lines of

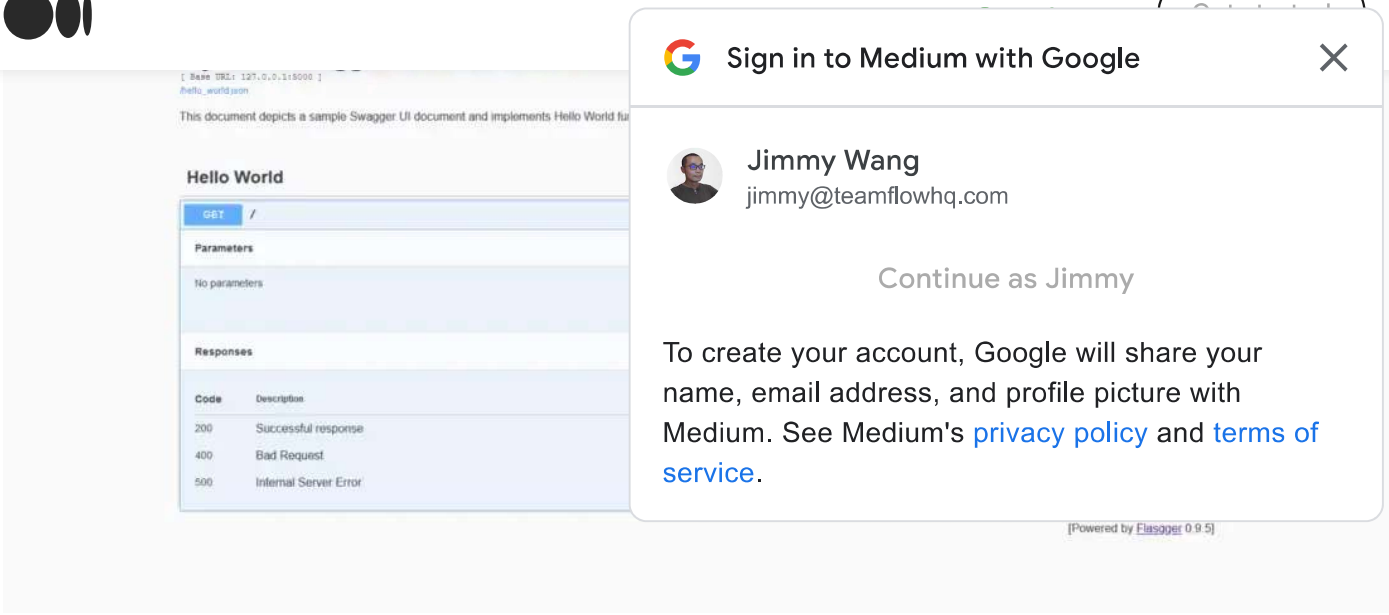
```
app.run()
```

The code when executed will run on `lo`
following document is displayed when



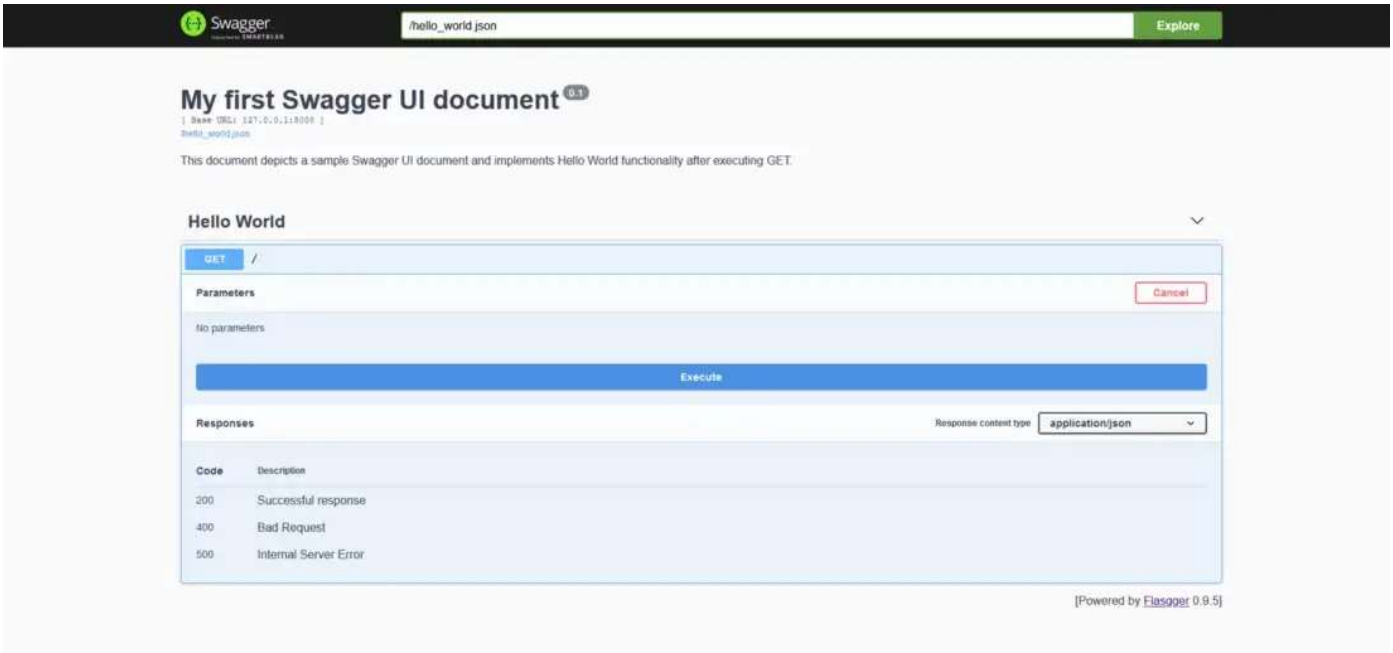
Document obtained after running 127.0.0.1:5000/apidocs in browser

As you can see, this document contains the title, version and description of the API as defined in the YAML file. Once the GET response is selected in the Swagger UI document, the document expand as:



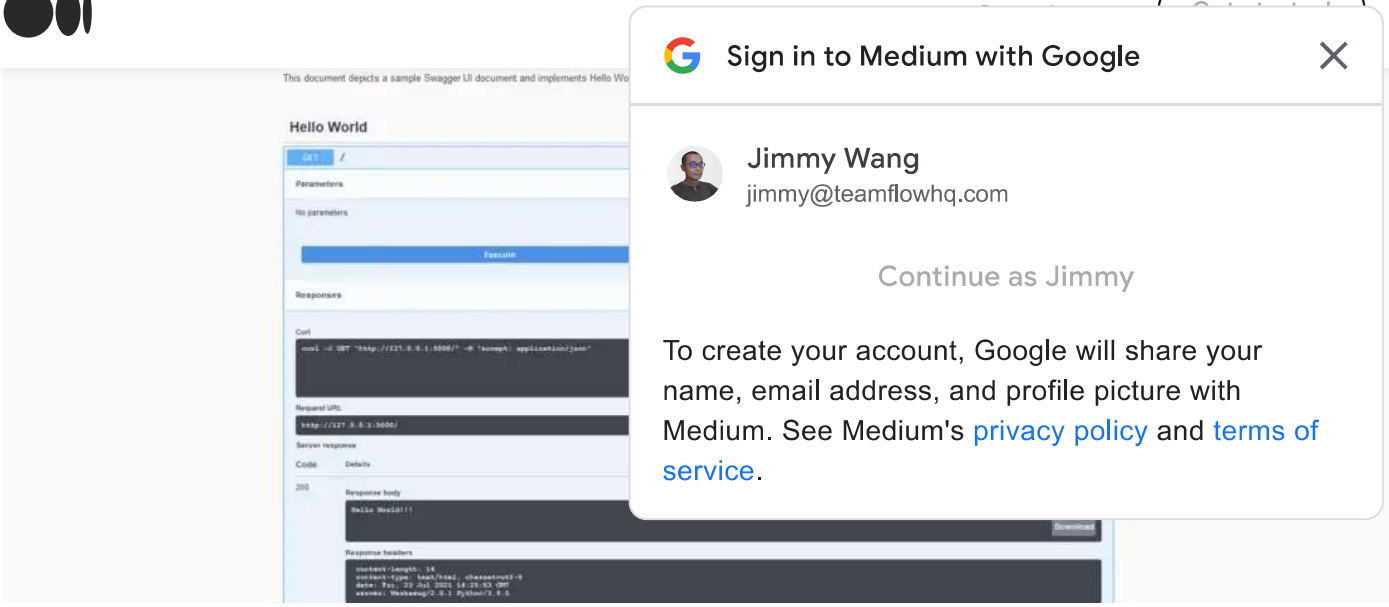
Result obtained after selecting GET response from the document

When you click the "**Try it out**" button, the document displays the "**Execute**" button as follows:



Result obtained after selecting "Try it out" button

Once the "**Execute**" option is selected, the `Hello World` output is displayed as shown below:-



Result after executing GET response by selecting “Execute” option

As seen, the API response is returned within the `“Response body”` of the GET method. The `“Response headers”` contain basic information of the response such as length of the response, type of the response, date and time at which the response was received (time will be displayed as per GMT time zone) etc.

Hope you have understood the significance of creating Swagger UI documents for APIs and how to use them as testing tools for the API. I would be happy to acknowledge any questions and doubts which can be posted through the comments.