flaskdoc

Release 0.2.0

Rowland Ogwara

Jun 14, 2021

CONTENTS

FlaskDoc is an extension of the puparley know to programmatically compose openapi specifications for flask endpoints as a part of code without needing to write a separate yaml file, and it comes with SwaggerUI embedded. Its main focus is on documentation which frees developers to focus on getting their services coded.

CHAPTER

ONE

USER'S GUIDE

This section provides documentation on how to get started using flaskdoc in your application.

1.1 Getting started

flaskdoc provides decorators that can be used to automatically generate openapi v3 specifications from existing flask routes. This sections describes how to get started quickly.

1.1.1 Step 1. Install

`flaskdoc` is available on PyPi,

1.1.2 Step 2. Configure

Register flaskdoc with your flask `app` instance. This requires top level openapi models like Info, Server, Tags etc. These should match your api's expectations. Flaskdoc exposes swagger specific models via `flaskdoc.swagger`

Note: default url is `/docs`, this can be customized by setting the *docs_path* parameter of *register_openapi*. You can also use Redoc instead of the default swagger ui, by setting the *use_redoc* parameter to True.

```
from flaskdoc import swagger

contact_info = swagger.Contact(
    name="Contact Name",
    email="Contact Email",
    url="Contact URL",
)
license_info = swagger.License(
    name="License Name",
    url="License URL",
)

info = swagger.Info(
    title="API title",
    version="API version",
    contact=contact_info,
```

(continues on next page)

(continued from previous page)

```
    license=license_info,
)
```

Registering openapi routes using openapi involves simply doing

Note: See *Snippets* section for more detailed examples on how to use the register_openapi function

```
import flask
from flaskdoc import register_openapi, swagger

app = flask.Flask("Fancy flask App)
info = swagger.Info(...)

register_openapi(app, info=info)
```

Under the hood, flaskdoc simply does:

```
ui.add_url_rule("/", view_func=register_docs_ui, methods=["GET"])
ui.add_url_rule("/<path:path>", view_func=register_docs_ui, methods=["GET"])
ui.add_url_rule("/openapi.json", view_func=register_json_path, methods=["GET"])
ui.add_url_rule("/openapi.yaml", view_func=register_yaml_path, methods=["GET"])
app.register_blueprint(ui, url_prefix=docs_path)
```

1.1.3 Step 3. Start Documenting

Again `flaskdoc` exposes http method decorators that can be used in documenting your api via `flaskdoc.swagger`.

Simple GET

```
blp = flask.Blueprint("Dummy", __name__, url_prefix="/v1")

@swagger.GET(
    tags=["getEcho"],
    operation_id="getEcho",
    parameters=[swagger.PathParameter(name="sample", schema=str)],
    description="Retrieve echos wit Get",
    responses={
        "200": swagger.ResponseObject(
            description="Success", content=jo.PlainText(schema=jo.Email()),
        )
    },
)
@blp.route("/echo/<string:sample>", methods=["GET"])
def echo(sample: str):
    """
    Sample GET request
    Returns: Echos back whatever was sent
```

(continues on next page)

(continued from previous page)

```
    """
    return sample
```

Simple POST

```
blp = flask.Blueprint("Dummy", __name__, url_prefix="/v1")
@swagger.POST(
    tags=["administrator"],
    description="Posts an Echo",
    responses={"201": swagger.ResponseObject(description="OK")},
)
@blp.route("/echo", methods=["POST"])
def post():
    req = flask.request.get_json(force=True)
    return flask.jsonify(req), 200
```

1.1.4 Data Models

Flaskdoc includes a framework agnostic way of automatically converting native python classes into swagger compatible json schema. See *Data Models* for more information.

```
import flask
from flaskdoc import swagger

ts = flask.Blueprint(__name__)

class OakTown:
""" Sample class without any special annotations """

    oaks = None
    smugs = 1  # type: int
    snux = "2"  # type: str


@swagger.GET(
  tags=["listOaks"],
  summary="Lists all towns with oaks",
  responses={
    "200": swagger.JsonType(schema=[OakTown])
  }
)
@ts.route("/list", methods=["GET"])
def list_oaks():
    return []


@swagger.POST(
  tags=["createTown],
  summary="Create a new OakTown",
```

(continues on next page)

(continued from previous page)

```
  request_body=swagger.RequestBody(
    content=[swagger.JsonType(schema=OakTown), swagger.XmlType(schema=OakTown)]
  )
)
@ts.route("", methods=["POST"]
def create_town():
    return OakTown())
```

1.2 Examples

`flaskdoc` includes example projects which was used extensively to test the usability of the project. To run examples `flaskdoc` needs to be installed as dev using

```
$ pip install flaskdoc[dev]
```

1.2.1 Running Examples

Flaskdoc comes with a basic command line tool for running examples that demonstrates the various capabilities of the project

Usage

flaskdoc examples can be invoked as follows:

```
$ flaskdoc start -n <example>
```

Where example can either be one of

- petstore
- inventory
- link-example
- api-with-example

1.2.2 Petstore Example

Implements the standard petstore.swagger.io specification provided by swagger. To run petstore example:

```
$ flaskdoc start -n petstore
```

1.2.3 Inventory Example

Implements the simple inventory api provided by swaggerhub. To run inventory example:

```
$ flaskdoc start -n inventory
```

Visit http://localhost:15172/docs to see the UI

1.3 Snippets

Code usage examples and snippets

1.3.1 Reusable Components

Reusable examples and other components can be adding to the components using:

```
from flaskdoc import swagger, register_openapi

examples = {
    "foo": swagger.Example(value=10)
}
links = {
    "l1": swagger.Link(operation_id="L!")
}

register_openapi(app, info, examples=examples, links=links)
```

1.3.2 Reference Objects

Resuable components can be referenced using proper ReferenceObject

```
@swagger.GET(
    tags=["getVersionDetailsv2"],
    summary="Show API versions details",
    responses={
        "200": swagger.ResponseObject(
            description="200 response",
            content=swagger.JsonType(examples={"foo": swagger.ExampleReference("v2-foo")}
↪),
        ),
        "300": swagger.ResponseObject(
            description="300 response",
            content=swagger.JsonType(examples={"foo": swagger.ExampleReference("v2-foo")}
↪),
        ),
    },
)
@api.route("/v2", methods=["GET"])
def details():
    return flask.jsonify(examples["v2-foo"])
```

1.3.3 Defining Examples

```
examples = {
    "foo": swagger.Example(
        value={
            "versions": [
                {
                    "status": "CURRENT",
                    "updated": "2011-01-21T11:33:21Z",
                    "id": "v2.0",
                    "links": [{"href": "http://127.0.0.1:8774/v2/", "rel": "self"}],
                },
                {
                    "status": "EXPERIMENTAL",
                    "updated": "2013-07-23T11:33:21Z",
                    "id": "v3.0",
                    "links": [{"href": "http://127.0.0.1:8774/v3/", "rel": "self"}],
                },
            ]
        }
    ),
    "v2-foo": swagger.Example(
        value={
            "version": {
                "status": "CURRENT",
                "updated": "2011-01-21T11:33:21Z",
                "media-types": [
                    {
                        "base": "application/xml",
                        "type": "application/vnd.openstack.compute+xml;version=2",
                    },
                    {
                        "base": "application/json",
                        "type": "application/vnd.openstack.compute+json;version=2",
                    },
                ],
                "id": "v2.0",
                "links": [
                    {"href": "http://23.253.228.211:8774/v2/", "rel": "self"},
                    {
                        "href": "http://docs.openstack.org/api/openstack-compute/2/os-
↪compute-devguide-2.pdf",
                        "type": "application/pdf",
                        "rel": "describedby",
                    },
                    {
                        "href": "http://docs.openstack.org/api/openstack-compute/2/wadl/
↪os-compute-2.wadl",
                        "type": "application/vnd.sun.wadl+xml",
                        "rel": "describedby",
                    },
                ],
            }
```

(continues on next page)

(continued from previous page)

```
        }
    ),
}
```

1.3.4 Using Enums

```
class RepositoryState(enum.Enum):

    open = "open"
    merged = "merged"
    declined = "declined"
```

```
)


get_pull_requests_by_repository = swagger.GET(
    operation_id="getPullRequestByRepository",
    parameters=[
        swagger.PathParameter(
            name="username",
            schema=str,
        ),
        swagger.PathParameter(name="slug", schema=str),
        swagger.QueryParameter(name="state", schema=schemas.RepositoryState),
    ],
    responses={
        "200": swagger.ResponseObject(
```

1.4 API Reference

Provides full reference to `flaskdoc's` API

1.4.1 Swagger Models

Swagger models implementations

class flaskdoc.swagger.models.**ApiKeySecurityScheme**(*name: str*, *description: Optional[str] = None*, *extensions={}*)

Bases: *flaskdoc.swagger.models.SecurityScheme*

OpenAPI security scheme definition with type apiKey

Example

Sample security requirements .. code-block:: json

{ “api_key”: []

}

property q_in

class flaskdoc.swagger.models.**AuthorizationCodeOAuthFlow**(*authorization_url*, *token_url*, *scopes*, *refresh_url=None*, *extensions=None*)

Bases: *flaskdoc.core.ExtensionMixin*

Authorization Code OAuth2 Flow

class flaskdoc.swagger.models.**Callback**(*expression:* flaskdoc.swagger.models.PathItem)

Bases: *flaskdoc.core.ModelMixin*

A map of possible out-of band callbacks related to the parent operation. Each value in the map is a Path Item Object that describes a set of requests that may be initiated by the API provider and the expected responses. The key value used to identify the callback object is an expression, evaluated at runtime, that identifies a URL to use for the callback operation.

class flaskdoc.swagger.models.**ClientCredentialsOAuthFlow**(*token_url*, *scopes*, *authorization_url=None*, *refresh_url=None*, *extensions=None*)

Bases: *flaskdoc.core.ExtensionMixin*

Client Credentials OAuth FLow

class flaskdoc.swagger.models.**ComponentType**(*value*)

Bases: enum.Enum

An enumeration.

CALLBACKS = 'callbacks'

EXAMPLE = 'examples'

HEADER = 'headers'

LINK = 'links'

PARAMETER = 'parameters'

REQUEST_BODY = 'request+bodies'

RESPONSE = 'responses'

SCHEMA = 'schemas'

SECURITY_SCHEME = 'security_schemes'

class flaskdoc.swagger.models.**Components**(*schemas: Optional[dict] = None*, *responses: Optional[dict] = None*, *parameters: Optional[dict] = None*, *examples: Optional[dict] = None*, *request_bodies: Optional[dict] = None*, *headers: Optional[dict] = None*, *security_schemes: Optional[dict] = None*, *links: Optional[dict] = None*, *callbacks: Optional[dict] = None*, *extensions={}*)

Bases: *flaskdoc.core.ExtensionMixin*

Holds a set of reusable objects for different aspects of the OAS.

All objects defined within the components object will have no effect on the API unless they are explicitly referenced from properties outside the components object.

PATTERN = re.compile('^[a-zA-Z0-9.-_]+$')

add_component(*component_type*, *components*)
Adds components

Parameters

- **component_type** (ComponentType) – type of component
- **components** (*dict[str, Any]*) – key value mapping of components

Raises **ValueError** – If key is not a valid value for regex `^[a-zA-Z0-9.-_]+$`

class flaskdoc.swagger.models.**Contact**(*name: Optional[str] = None*, *email: Optional[str] = None*, *url: Optional[str] = None*, *extensions={}*)
Bases: *flaskdoc.core.ExtensionMixin*

Contact information for the exposed API.

This object MAY be extended with Specification Extensions.

Properties: name: The identifying name of the contact person/organization. email: The email address of the contact person/organization. MUST be in the format of an email address. url: The URL pointing to the contact information. MUST be in the format of a URL.

validate(*_*, *url*)
Validates the name of all provided extensions

class flaskdoc.swagger.models.**ContainerModel**(*items={}*)
Bases: *flaskdoc.core.ModelMixin*

add(*key*, *item*)
Adds an item :param key: item key :type key: str :param item: item :type item: dict

get(*key*)

to_dict()
Converts object to dictionary

class flaskdoc.swagger.models.**CookieParameter**(*name: str*, *required=None*, *description: Optional[str] = None*, *deprecated=None*, *allow_empty_value=None*, *allow_reserved=None*, *schema=None*, *content=None*, *explode=None*, *example=None*, *examples: Optional[dict] = None*, *extensions={}*)
Bases: *flaskdoc.swagger.models.Parameter*

class flaskdoc.swagger.models.**DELETE**(*responses: dict*, *tags: Optional[list] = None*, *summary: Optional[str] = None*, *description: Optional[str] = None*, *external_docs: Optional[flaskdoc.swagger.models.ExternalDocumentation] = None*, *operation_id: Optional[str] = None*, *parameters: Optional[list] = None*, *request_body=None*, *callbacks: Optional[flaskdoc.swagger.models.SwaggerDict] = None*, *deprecated=None*, *security: Optional[list] = None*, *servers: Optional[list] = None*, *extensions={}*)
Bases: *flaskdoc.swagger.models.Operation*

property http_method

class flaskdoc.swagger.models.**ExampleReference**(*ref: str*)
Bases: *flaskdoc.swagger.models.ReferenceObject*

class flaskdoc.swagger.models.**ExternalDocumentation**(*url: str*, *description: Optional[str] = None*, *extensions={}*)
Bases: *flaskdoc.core.ExtensionMixin*

Allows referencing an external resource for extended documentation.

class flaskdoc.swagger.models.**GET**(*responses: dict*, *tags: Optional[list] = None*, *summary: Optional[str] = None*, *description: Optional[str] = None*, *external_docs: Optional[*flaskdoc.swagger.models.ExternalDocumentation*] = None*, *operation_id: Optional[str] = None*, *parameters: Optional[list] = None*, *request_body=None*, *callbacks: Optional[*flaskdoc.swagger.models.SwaggerDict*] = None*, *deprecated=None*, *security: Optional[list] = None*, *servers: Optional[list] = None*, *extensions={}*)
Bases: *flaskdoc.swagger.models.Operation*

property http_method

class flaskdoc.swagger.models.**HEAD**(*responses: dict*, *tags: Optional[list] = None*, *summary: Optional[str] = None*, *description: Optional[str] = None*, *external_docs: Optional[*flaskdoc.swagger.models.ExternalDocumentation*] = None*, *operation_id: Optional[str] = None*, *parameters: Optional[list] = None*, *request_body=None*, *callbacks: Optional[*flaskdoc.swagger.models.SwaggerDict*] = None*, *deprecated=None*, *security: Optional[list] = None*, *servers: Optional[list] = None*, *extensions={}*)
Bases: *flaskdoc.swagger.models.Operation*

property http_method

class flaskdoc.swagger.models.**Header**(*required=None*, *description: Optional[str] = None*, *deprecated=None*, *allow_empty_value=None*, *allow_reserved=None*, *schema=None*, *content=None*, *explode=None*, *example=None*, *examples: Optional[dict] = None*, *extensions={}*)
Bases: *flaskdoc.swagger.models.HeaderParameter*

class flaskdoc.swagger.models.**HeaderParameter**(*name: str*, *required=None*, *description: Optional[str] = None*, *deprecated=None*, *allow_empty_value=None*, *allow_reserved=None*, *schema=None*, *content=None*, *explode=None*, *example=None*, *examples: Optional[dict] = None*, *extensions={}*)
Bases: *flaskdoc.swagger.models.Parameter*

class flaskdoc.swagger.models.**HttpMethod**(*value*)
Bases: enum.Enum

An enumeration.

DELETE = 'DELETE'

GET = 'GET'

HEAD = 'HEAD'

OPTIONS = 'OPTIONS'

PATCH = 'patch'

POST = 'POST'

PUT = 'PUT'

TRACE = 'TRACE'

class flaskdoc.swagger.models.**HttpSecurityScheme**(*scheme: string*, *bearer_format='bearer'*, *description: str = None*, *extensions={}*)

Bases: *flaskdoc.swagger.models.SecurityScheme*

OpenAPI security scheme definition with type http

class flaskdoc.swagger.models.**ImplicitOAuthFlow**(*authorization_url*, *scopes*, *token_url=None*, *refresh_url=None*)

Bases: *flaskdoc.core.ExtensionMixin*

Implicit OAuth2 Flow

class flaskdoc.swagger.models.**Info**(*title: str*, *version: str*, *description: Optional[str] = None*, *terms_of_service: Optional[str] = None*, *contact: Optional[*flaskdoc.swagger.models.Contact*] = None*, *license: Optional[*flaskdoc.swagger.models.License*] = None*, *extensions={}*)

Bases: *flaskdoc.core.ExtensionMixin*

The object provides metadata about the API.

The metadata MAY be used by the clients if needed, and MAY be presented in editing or documentation generation tools for convenience. This object MAY be extended with Specification Extensions.

Properties: title: REQUIRED. The title of the API. version: REQUIRED. The version of the OpenAPI document (which is distinct from the OpenAPI Specification

version or the API implementation version).

description: A short description of the API. CommonMark syntax MAY be used for rich text representation. terms_of_service: A URL to the Terms of Service for the API. MUST be in the format of a URL. contact: The contact information for the exposed API. license: The license information for the exposed API.

class flaskdoc.swagger.models.**License**(*name: str*, *url: Optional[str] = None*, *extensions={}*)

Bases: *flaskdoc.core.ExtensionMixin*

License information for the exposed API.

This object MAY be extended with Specification Extensions.

Properties: name: REQUIRED. The license name used for the API. url: A URL to the license used for the API. MUST be in the format of a URL.

validate(*_*, *url*)

Validates the name of all provided extensions

class flaskdoc.swagger.models.**Link**(*operation_ref: Optional[str] = None*, *operation_id: Optional[str] = None*, *description: Optional[str] = None*, *parameters: Optional[*flaskdoc.swagger.models.SwaggerDict*] = None*, *request_body=None*, *server: Optional[*flaskdoc.swagger.models.Server*] = None*, *extensions={}*)

Bases: *flaskdoc.core.ExtensionMixin*

The Link object represents a possible design-time link for a response. The presence of a link does not guarantee the caller's ability to successfully invoke it, rather it provides a known relationship and traversal mechanism between responses and other operations. Unlike dynamic links (i.e. links provided in the response payload), the OAS linking mechanism does not require link information in the runtime response. For computing links, and

providing instructions to execute them, a runtime expression is used for accessing values in an operation and using them as parameters while invoking the linked operation.

class flaskdoc.swagger.models.**LinkReference**(*ref: str*)
Bases: *flaskdoc.swagger.models.ReferenceObject*

class flaskdoc.swagger.models.**OAuth2SecurityScheme**(*flows*, *extensions={}*)
Bases: *flaskdoc.swagger.models.SecurityScheme*

OpenAPI security scheme definition with type oauth2

class flaskdoc.swagger.models.**OAuthFlow**(*authorization_url: str*, *token_url: str*, *refresh_url: str*, *scopes: {}*, *extensions={}*)
Bases: *flaskdoc.core.ExtensionMixin*

Configuration details for a supported OAuth Flow

class flaskdoc.swagger.models.**OPTIONS**(*responses: dict*, *tags: Optional[list] = None*, *summary: Optional[str] = None*, *description: Optional[str] = None*, *external_docs: Optional[*flaskdoc.swagger.models.ExternalDocumentation*] = None*, *operation_id: Optional[str] = None*, *parameters: Optional[list] = None*, *request_body=None*, *callbacks: Optional[*flaskdoc.swagger.models.SwaggerDict*] = None*, *deprecated=None*, *security: Optional[list] = None*, *servers: Optional[list] = None*, *extensions={}*)
Bases: *flaskdoc.swagger.models.Operation*

property http_method

class flaskdoc.swagger.models.**OpenApi**(*info*, *paths*, *version='3.0.3'*, *tags=None*, *servers=None*, *external_docs=None*, *components=None*)
Bases: *flaskdoc.core.ModelMixin*

This is the root document object of the OpenAPI document.

OpenApi specs tree, contains the overall specs for the API Properties:

openapi (str): Open API version used by API info (flaskdoc.swagger.info.Info): open api info object paths (Paths): Paths definitions

add_paths(*paths*, *url_prefix=None*, *blp_prefix=None*)
Updates paths to include all paths in *paths* :param paths: :param url_prefix: prefix :type url_prefix: str :param blp_prefix: blueprint url prefix :type blp_prefix: str

Returns:

add_server(*server*)

add_tag(*tag*)
Adds a tag to the top level spec :param tag: tag to add :type tag: swagger.Tag

class flaskdoc.swagger.models.**OpenIDConnectScheme**(*open_id_connect_url: str*, *extensions={}*)
Bases: *flaskdoc.swagger.models.SecurityScheme*

OpenAPI security scheme definition with type openidConnect

validate(*_*, *url*)
Validates the name of all provided extensions

class flaskdoc.swagger.models.**Operation**(*responses: dict*, *tags: Optional[list] = None*, *summary: Optional[str] = None*, *description: Optional[str] = None*, *external_docs: Optional[flaskdoc.swagger.models.ExternalDocumentation] = None*, *operation_id: Optional[str] = None*, *parameters: Optional[list] = None*, *request_body=None*, *callbacks: Optional[flaskdoc.swagger.models.SwaggerDict] = None*, *deprecated=None*, *security: Optional[list] = None*, *servers: Optional[list] = None*, *extensions={}*)

Bases: *flaskdoc.core.ExtensionMixin*, *flaskdoc.core.ApiDecoratorMixin*

Describes a single API operation on a path.

add_parameter(*parameter: Union[flaskdoc.swagger.models.Parameter, flaskdoc.swagger.models.ReferenceObject]*)

static from_op(*http_method: str*, *responses:* flaskdoc.swagger.models.SwaggerDict)

Factory for creating instances of Http Operations

property http_method

class flaskdoc.swagger.models.**PATCH**(*responses: dict*, *tags: Optional[list] = None*, *summary: Optional[str] = None*, *description: Optional[str] = None*, *external_docs: Optional[flaskdoc.swagger.models.ExternalDocumentation] = None*, *operation_id: Optional[str] = None*, *parameters: Optional[list] = None*, *request_body=None*, *callbacks: Optional[flaskdoc.swagger.models.SwaggerDict] = None*, *deprecated=None*, *security: Optional[list] = None*, *servers: Optional[list] = None*, *extensions={}*)

Bases: *flaskdoc.swagger.models.Operation*

property http_method

class flaskdoc.swagger.models.**POST**(*responses: dict*, *tags: Optional[list] = None*, *summary: Optional[str] = None*, *description: Optional[str] = None*, *external_docs: Optional[flaskdoc.swagger.models.ExternalDocumentation] = None*, *operation_id: Optional[str] = None*, *parameters: Optional[list] = None*, *request_body=None*, *callbacks: Optional[flaskdoc.swagger.models.SwaggerDict] = None*, *deprecated=None*, *security: Optional[list] = None*, *servers: Optional[list] = None*, *extensions={}*)

Bases: *flaskdoc.swagger.models.Operation*

property http_method

class flaskdoc.swagger.models.**PUT**(*responses: dict*, *tags: Optional[list] = None*, *summary: Optional[str] = None*, *description: Optional[str] = None*, *external_docs: Optional[flaskdoc.swagger.models.ExternalDocumentation] = None*, *operation_id: Optional[str] = None*, *parameters: Optional[list] = None*, *request_body=None*, *callbacks: Optional[flaskdoc.swagger.models.SwaggerDict] = None*, *deprecated=None*, *security: Optional[list] = None*, *servers: Optional[list] = None*, *extensions={}*)

Bases: *flaskdoc.swagger.models.Operation*

property http_method

class flaskdoc.swagger.models.**Parameter**(*name: str*, *required=None*, *description: Optional[str] = None*, *deprecated=None*, *allow_empty_value=None*, *allow_reserved=None*, *schema=None*, *content=None*, *explode=None*, *example=None*, *examples: Optional[dict] = None*, *extensions={}*)

Bases: *flaskdoc.core.ExtensionMixin*, *flaskdoc.core.ApiDecoratorMixin*

Describes a single operation parameter. A unique parameter is defined by a combination of a name and location.

merge(*parameter*)

property q_in

property q_style

class flaskdoc.swagger.models.**ParameterLocation**(*value*)

Bases: enum.Enum

An enumeration.

COOKIE = 'cookie'

HEADER = 'header'

PATH = 'path'

QUERY = 'query'

class flaskdoc.swagger.models.**PasswordOAuthFlow**(*token_url*, *scopes*, *authorization_url=None*, *refresh_url=None*, *extensions=None*)

Bases: *flaskdoc.core.ExtensionMixin*

Password based OAuth2 Flow

class flaskdoc.swagger.models.**PathItem**(*ref: Optional[str] = None*, *summary: Optional[str] = None*, *description: Optional[str] = None*, *servers: Optional[list] = None*, *parameters: Optional[list] = None*, *get: Optional[*flaskdoc.swagger.models.Operation*] = None*, *delete: Optional[*flaskdoc.swagger.models.Operation*] = None*, *head: Optional[*flaskdoc.swagger.models.Operation*] = None*, *options: Optional[*flaskdoc.swagger.models.Operation*] = None*, *patch: Optional[*flaskdoc.swagger.models.Operation*] = None*, *post: Optional[*flaskdoc.swagger.models.Operation*] = None*, *put: Optional[*flaskdoc.swagger.models.Operation*] = None*, *trace: Optional[*flaskdoc.swagger.models.Operation*] = None*, *extensions={}*)

Bases: *flaskdoc.core.ExtensionMixin*

Describes the operations available on a single path. A Path Item MAY be empty, due to ACL constraints. The path itself is still exposed to the documentation viewer but they will not know which operations and parameters are available.

add_operation(*operation*)

Adds an operation :param operation: operation to add :type operation: Operation

add_parameter(*parameter*)

add_server(*server*)

Adds an alternative server to service all operations in this path.

Parameters **server** (Server) – alternative server to add

merge_path_item(*path_item*)
Merges another path item into this on :param path_item: PathItem instance to merge :type path_item: PathItem

class flaskdoc.swagger.models.**PathParameter**(*name: str*, *description: Optional[str] = None*, *deprecated=None*, *allow_empty_value=None*, *allow_reserved=None*, *schema=None*, *content=None*, *explode=None*, *example=None*, *examples: Optional[dict] = None*, *extensions={}*)
Bases: *flaskdoc.swagger.models.Parameter*

class flaskdoc.swagger.models.**Paths**(*items={}*)
Bases: *flaskdoc.swagger.models.ContainerModel*

Holds the relative paths to the individual endpoints and their operations. The path is appended to the URL from the Server Object in order to construct the full URL. The Paths MAY be empty, due to ACL constraints.

add(*relative_url*, *path_item*)
Adds a path item :param relative_url: path url name, eg */echo* :type relative_url: str :param path_item: PathItem instance describing the path :type path_item: PathItem|SwaggerDict

class flaskdoc.swagger.models.**QueryParameter**(*name: str*, *required=None*, *description: Optional[str] = None*, *deprecated=None*, *allow_empty_value=None*, *allow_reserved=None*, *schema=None*, *content=None*, *explode=None*, *example=None*, *examples: Optional[dict] = None*, *extensions={}*)
Bases: *flaskdoc.swagger.models.Parameter*

class flaskdoc.swagger.models.**ReferenceObject**(*ref: str*)
Bases: *flaskdoc.core.ModelMixin*

to_dict()
Converts object to dictionary

class flaskdoc.swagger.models.**RelativePath**(*url: str*, *len: int = 0*, *params={}*)
Bases: `object`

parse(*url_template*)
Extracts positional parameters from url :param url_template:

Returns:

class flaskdoc.swagger.models.**RequestBody**(*content*, *description: Optional[str] = None*, *required=None*, *extensions={}*)
Bases: `flaskdoc.swagger.schema.ContentMixin`, *flaskdoc.core.ExtensionMixin*

class flaskdoc.swagger.models.**ResponseObject**(*description: str*, *content: Optional[*flaskdoc.swagger.models.SwaggerDict*] = None*, *headers: Optional[*flaskdoc.swagger.models.SwaggerDict*] = None*, *links: Optional[*flaskdoc.swagger.models.SwaggerDict*] = None*, *extensions={}*)
Bases: `flaskdoc.swagger.schema.ContentMixin`, *flaskdoc.core.ExtensionMixin*

Describes a single response from an API Operation, including design-time, static links to operations based on the response.

add_header(*name: str*, *header: Union[*flaskdoc.swagger.models.ReferenceObject, flaskdoc.swagger.models.HeaderParameter*]*)

add_link(*link_name: str*, *link: Union[*flaskdoc.swagger.models.Link, flaskdoc.swagger.models.ReferenceObject*]*)

class flaskdoc.swagger.models.**ResponsesObject**(*default: Optional[*flaskdoc.swagger.models.ResponseObject*] = None*, *responses: Optional[dict] = None*, *extensions={}*)

Bases: *flaskdoc.core.ExtensionMixin*

A container for the expected responses of an operation. The container maps a HTTP response code to the expected response. The documentation is not necessarily expected to cover all possible HTTP response codes because they may not be known in advance. However, documentation is expected to cover a successful operation response and any known errors. The default MAY be used as a default response object for all HTTP codes that are not covered individually by the specification. The Responses Object MUST contain at least one response code, and it SHOULD be the response for a successful operation call.

add_response(*status_code: str*, *response:* flaskdoc.swagger.models.ResponseObject)

class flaskdoc.swagger.models.**SecurityScheme**

Bases: *flaskdoc.core.ExtensionMixin*

Defines a security scheme that can be used by the operations.

Supported schemes are HTTP authentication, an API key (either as a header or as a query parameter), OAuth2's common flows (implicit, password, application and access code) as defined in RFC6749, and OpenID Connect Discovery.

property q_type

class flaskdoc.swagger.models.**SecuritySchemeType**(*value*)

Bases: enum.Enum

An enumeration.

API_KEY = 'apiKey'

HTTP = 'http'

OAUTH2 = 'oauth2'

OPEN_ID_CONNECT = 'openIdConnect'

class flaskdoc.swagger.models.**Server**(*url: str*, *description: Optional[str] = None*, *variables: Optional[dict] = None*, *extensions={}*)

Bases: *flaskdoc.core.ExtensionMixin*

An object representing a Server.

This object MAY be extended with Specification Extensions.

Properties:

url: REQUIRED. A URL to the target host. This URL supports Server Variables and MAY be relative, to indicate that the host location is relative to the location where the OpenAPI document is being served. Variable substitutions will be made when a variable is named in {brackets}.

description: An optional string describing the host designated by the URL. CommonMark syntax MAY be used for rich text representation.

variables: A map between a variable name and its value. The value is used for substitution in the server's URL template.

add_variable(*name: str*, *variable:* flaskdoc.swagger.models.ServerVariable)

Adds a server variable :param name: variable name :param variable: Server variable instance

class flaskdoc.swagger.models.**ServerVariable**(*default: str*, *enum: Optional[list] = None*, *description: Optional[str] = None*, *extensions={}*)

Bases: *flaskdoc.core.ExtensionMixin*

An object representing a Server Variable for server URL template substitution.

This object MAY be extended with Specification Extensions.

Properties:

default: REQUIRED. The default value to use for substitution, which SHALL be sent if an alternate value is not supplied. Note this behavior is different than the Schema Object's treatment of default values, because in those cases parameter values are optional.

enum: An enumeration of string values to be used if the substitution options are from a limited set description: An optional description for the server variable. CommonMark syntax MAY be used for rich text

representation.

class flaskdoc.swagger.models.**Style**(*value*)

Bases: enum.Enum

Style values defined to aid serializing different simple parameters

DEEP_OBJECT = 'deepObject'

FORM = 'form'

LABEL = 'label'

MATRIX = 'matrix'

PIPE_DELIMITED = 'pipeDelimited'

SIMPLE = 'simple'

SPACE_DELIMITED = 'spaceDelimited'

class flaskdoc.swagger.models.**SwaggerDict**

Bases: collections.OrderedDict, *flaskdoc.core.DictMixin*

Used to filter out properties that are not set

class flaskdoc.swagger.models.**TRACE**(*responses: dict*, *tags: Optional[list] = None*, *summary: Optional[str] = None*, *description: Optional[str] = None*, *external_docs: Optional[*flaskdoc.swagger.models.ExternalDocumentation*] = None*, *operation_id: Optional[str] = None*, *parameters: Optional[list] = None*, *request_body=None*, *callbacks: Optional[*flaskdoc.swagger.models.SwaggerDict*] = None*, *deprecated=None*, *security: Optional[list] = None*, *servers: Optional[list] = None*, *extensions={}*)

Bases: *flaskdoc.swagger.models.Operation*

property http_method

class flaskdoc.swagger.models.**Tag**(*name: str*, *description: Optional[str] = None*, *external_docs: Optional[*flaskdoc.swagger.models.ExternalDocumentation*] = None*, *extensions={}*)

Bases: *flaskdoc.core.ExtensionMixin*, *flaskdoc.core.ApiDecoratorMixin*

external_doc(*url*, *description=None*)

flaskdoc.swagger.validators module

flaskdoc.swagger.validators.**validate_url**(*url: str*, *label: str*)

1.4.2 Data Models

flaskdoc introspects native python objects and converts then into swagger appropriate json schema. Data models can be defined using either standard python classes, dataclasses or attrs. Flaskdoc will automatically convert them into json schema.

Schema from Python Classes

Note: For native python classes like the example above, default values are used to decipher the types, else it defaults to string. Finer control over types can be achieved using either python data classes or attrs or type annotations

```
from flaskdoc.schema import SchemaFactory

class OakTown:
""" Sample class without any special annotations """

    oaks = None
    smugs = 1  # type: int
    snux = "2"  # type: str

factory = SchemaFactory()
schema = factory.get_schema(OakTown)
print(schema.json())
# {'$ref': '#/components/schemas/OakTown', 'description': 'Sample class without any special␣
↪annotations '}
print(factory.schemas['OakTown'])
# {
#   "properties": {
#     "oaks": {
#       "type": "string"
#     },
#     "smugs": {
#       "type": "integer"
#     },
#     "snux": {
#       "type": "string"
#     }
#   },
#   "type": "object"
# }
```

Schema using typing module

An example schema two array properties `samples` and `squeezes` and an integer property `density`

```
class SoakedBean(object):

    density: int = None
    samples: List[Sample] = []
    squeezes: Set[Squeezed] = {}
```

Schema using attr classes

Note: With `attr` types are deciphered using the `type` parameter of `attr.ib` if available, else if defaults to the type of the default value. If that is also not available it defaults to string.

```
from typing import List, Set

@attr.s
class Sample(object):
    """ Class with mixed attribute definitions """

    danni = "fear"
    palo = attr.ib(type=int)
    soap = attr.ib(type=SoapStar)
    hulu = attr.ib(default="NoNo")

class Squeezed:
""" Sample class with typed annotations """

    sample = 1
    spaces = 6


class SoakedBean(object):

    density: int = None
    samples: List[Sample] = []
    squeezes: Set[Squeezed] = {}

schema = factory.get_schema(SoakedBean)
print(schema.json())
# {"$ref": "#/components/schemas/SoakedBean"}

print(factory.schemas['SoakedBean'])
# {
#  "properties": {
#    "density": {
#      "type": "integer",
#      "format": "int32"
#    },
```

(continues on next page)

(continued from previous page)

```
#    "samples": {
#      "items": {
#        "$ref": "#/components/schemas/Sample",
#        "description": "Class with mixed attribute definitions "
#      },
#      "type": "array"
#    },
#    "squeezes": {
#      "items": {
#        "$ref": "#/components/schemas/Squeezed",
#        "description": "Sample class with typed annotations "
#      },
#      "type": "array"
#    }
#  },
#  "type": "object"
# }

print(factory.schemas['Sample'])
# {
#   "properties": {
#    "palo": {
#      "type": "integer",
#      "format": "int32"
#    },
#    "soap": {
#      "$ref": "#/components/schemas/SoapStar",
#      "description": "Simple attr based class "
#    },
#    "hulu": {
#      "type": "string"
#    },
#    "danni": {
#      "type": "string"
#    }
#  },
#  "type": "object"
# }
```

Schema with dataclasses (>=py3.7)

```
@dataclass
class SoakedBean(object):

    density: int = None
    samples: List[Sample] = []
    squeezes: Set[Squeezed] = {}
```

jo (json objects) models

Note: A drawback with defining schemas using the techniques described above is that there is no way to specify property constraints, eg max value for ints or minlength for strings.

jo is part of flaskdoc builtin functions that allows for adding constraints to native python classes. It makes it possible to define complex json schema representation from simple/plain/native python data types. It wraps around *attr* to enable developer provided schema constraints to properties in classes.

```
from flaskdoc import jo

@jo.schema(xml="SoakedBean")
class SoakedBean(object):

    density = jo.integer(default=10, minimum=9, maximum=10)
    samples = jo.array(item=Sample, min_items=1, xml="Samples")
    squeezes = jo.array(item=Squeezed, min_items=1, unique_items=True, xml="Squeezes")

# models can be used as normal data objects
bean = SoakedBean(density=12, samples=[Sample()], squeezes=[Squeezed()])
```

Model properties can be automatically converted to camel case by setting the camel_case_fields property of *jo.schema*
.. automodule:: flaskdoc.jo

members

1.4.3 core

Internal only functions and classes used by both swagger and flask specific customizations

class flaskdoc.core.**ApiDecoratorMixin**
Bases: object

Makes a model a decorator that registers itself

class flaskdoc.core.**DictMixin**
Bases: object

General usage mixin for handling nested dictionary conversion.

to_dict()
Converts object to dictionary

class flaskdoc.core.**ExtensionMixin**
Bases: *flaskdoc.core.ModelMixin*

add_extension(*name*, *value*)
Allows extensions to the Swagger Schema.

The field name MUST begin with x-, for example, x-internal-id. The value can be null, a primitive, an array or an object. :param name: custom extension name, must begin with x- :type name: str :param value: value, can be None, any object or list :type value: Any

Returns for chaining

Return type *ModelMixin*

Raises **ValueError** – if key name is invalid

validate(*_*, *ext*)
Validates the name of all provided extensions

static validate_extension_name(*value*)
Validates a custom extension name :param value: custom extension name :type value: str

Raises **ValueError** – if key name is invalid

class flaskdoc.core.**ModelMixin**
Bases: *flaskdoc.core.DictMixin*

Swagger Model mixin that provides common methods like to dict and to json

flaskdoc.core.**camel_case**(*snake_case*)
Converts snake case strings to camel case

Parameters **snake_case** (*str*) – raw snake case string, eg *sample_text*

Returns camel cased string

Return type str

1.5 Changelog

1.5.1 0.1.0

- First release.

CHAPTER

TWO

QUICKSTART

Install flaskdoc via `pip` or add to your project requirements/dependency

```
$ pip install flaskdoc
```

2.1 Register OpenAPI

Add top level openapi objects like Info, Contact, License etc

```
import flask
from flaskdoc import register_openapi, swagger

app = flask.Flask()
# initialize app, add all the blueprints you care about

# Create top level OpenAPI objects
# the info object
info = swagger.Info(
    title="Test",
    version="1.2.2",
    contact=swagger.Contact(
        name="Rowland", email="r.ogwara@gmail.com", url="https://github.com/kulgan"
    ),
    license=swagger.License(name="Apache 2.0", url="https://www.example.com/license"),
)

# servers names and variables if necessary
servers = [swagger.Server(url="http://localhost:15172")]

# top level tags
tags = [
    swagger.Tag(name="admin", description="Secured Admin-Only calls"),
    swagger.Tag(name="developers", description="Operations available to regular␣
↪developers"),
]

security_schemes = {
    "api_key": swagger.ApiKeySecurityScheme(name="api_key"),
}
```

(continues on next page)

(continued from previous page)

```
# register spec
register_openapi(app, info=info, servers=servers, tags=tags, security=security_schemes)
```

This adds the following endpoints to your list

- /docs
- /docs/openapi.yaml
- /docs/openapi.json

2.2 Start Documenting

Now start documenting you flask routes

```
blp = flask.Blueprint("Dummy", __name__, url_prefix="/v1")
@swagger.POST(
    tags=["administrator"],
    description="Posts an Echo",
    responses={"201": swagger.ResponseObject(description="OK")},
)
@blp.route("/echo", methods=["POST"])
def post():
    req = flask.request.get_json(force=True)
    return flask.jsonify(req), 200
```

A GET example with path parameter

```
blp = flask.Blueprint("Dummy", __name__, url_prefix="/v1")

@swagger.GET(
    tags=["getEcho"],
    operation_id="getEcho",
    parameters=[swagger.PathParameter(name="sample", schema=str)],
    description="Retrieve echos wit Get",
    responses={
        "200": swagger.ResponseObject(
            description="Success", content=jo.PlainText(schema=jo.Email()),
        )
    },
)
@blp.route("/echo/<string:sample>", methods=["GET"])
def echo(sample: str):
    """
    Sample GET request
    Returns: Echos back whatever was sent

    """
    return sample
```

Run your app and visit */docs* to see the generated openapi specs.

CHAPTER

THREE

CONTRIBUTING

Don't hesitate to create a Github issue for any **bugs** or **suggestions**.

CHAPTER

FOUR

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

f

INDEX

G

H

I

L

M

O

P

Q

R

S

T

V