sphinxcontrib-openapi

Oct 05, 2019

Contents

Hint: Check out sphinxcontrib-redoc if you are interested in separate three-panel OpenAPI spec rendering.

sphinxcontrib-openapi is a Sphinx extension to generate APIs docs from OpenAPI (fka Swagger) spec. It depends on sphinxcontrib-httpdomain that provides an HTTP domain for describing RESTful HTTP APIs, so we don't need to reinvent the wheel.

```
pip install sphinxcontrib-openapi
```

CHAPTER 1

How To Use?

Consider you have the following OpenAPI spec saved at `specs/openapi.yml`:

```yaml
swagger: "2.0"
info:
  title: Batcomputer API
  version: "1.0.0"
host: api.batcomputer.com
paths:
  /persons:
    get:
      summary: List Persons
      description: |
        Retrieves a list of all persons on file in the bat computer.
      responses:
        200:
          description: An array of Persons
          schema:
            type: array
            items:
              $ref: '#/definitions/Person'
  /evidence:
    get:
      summary: List Evidence
      description: |
        Retrieves a list of evidence ever found by world's greatest
        detective and his family.
      parameters:
        - name: marker
          in: query
          type: integer
          minimum: -1
          default: -1
          format: int64
          required: false
```

(continues on next page)

(continued from previous page)

```
          description: |
            The id of the last seen evidence. It's used for pagination purpose
            by returning 'limit' number of evidence after this one.
        - name: limit
          in: query
          type: integer
          format: int32
          minimum: 1
          maximum: 1000
          default: 20
          required: false
          description: |
            The maximum number of evidence to be returned by the query.
      responses:
        200:
          description: An array of evidence.
          schema:
            type: array
            items:
              $ref: '#/definitions/Evidence'
    post:
      summary: Create an Evidence
      description: |
        Creates a new evidence record in database.
      parameters:
        - name: evidence
          in: body
          schema:
            $ref: '#/definitions/Evidence'
      responses:
        201:
          description: An evidence.
          schema:
            $ref: '#/definitions/Evidence'
  /evidence/{id}:
    get:
      summary: Show Requested Evidence
      description: |
        Queries and returns an evidence with a passed ID.
      parameters:
        - name: id
          in: path
          type: integer
          format: int64
          required: true
          description: |
            A unique evidence identifier to query.
        - name: If-None-Match
          in: header
          type: string
          description: |
            Executes a request only if passed ETag doesn't equal to current
            resource one (i.e. resource has been changed).
      responses:
        200:
          description: An evidence.
          schema:
```

(continues on next page)

(continued from previous page)

```
            $ref: '#/definitions/Evidence'
          headers:
            ETag:
              description: |
                Entity tag is part of HTTP provided for web cache validation
                problem, but also used for optimistic concurrency control.
              type: string
        404:
          description: Evidence not found.
          schema:
            $ref: '#/definitions/Error'
definitions:
  Evidence:
    type: object
    properties:
      id:
        type: integer
        format: int64
        description: A unique evidence identifier.
      case:
        type: string
        description: A case when the evidence is found.
      data:
        type: string
        format: binary
        description: An evidence itself.
  Error:
    type: object
    properties:
      code:
        type: string
        description: A uniqe identifier of error.
      message:
        type: string
        description: A human readable error message.
  Person:
    type: object
    properties:
      id:
        type: integer
        format: int64
        description: Unique ID for a person
      name:
        type: string
        description: Name of a person
```

You can render it by using the `openapi` directive:

```
.. openapi:: specs/openapi.yml
```

and it will be rendered into something like:

GET /persons

List Persons

Retrieves a list of all persons on file in the bat computer.

Status Codes

- 200 OK – An array of Persons

Response JSON Object

- **[].id** (*integer*) – Unique ID for a person
- **[].name** (*string*) – Name of a person

GET /evidence
List Evidence

Retrieves a list of evidence ever found by world's greatest detective and his family.

Query Parameters

- **marker** (*integer*) – The id of the last seen evidence. It's used for pagination purpose by returning 'limit' number of evidence after this one.
- **limit** (*integer*) – The maximum number of evidence to be returned by the query.

Status Codes

- 200 OK – An array of evidence.

Response JSON Object

- **[].case** (*string*) – A case when the evidence is found.
- **[].data** (*string*) – An evidence itself.
- **[].id** (*integer*) – A unique evidence identifier.

POST /evidence
Create an Evidence

Creates a new evidence record in database.

Request JSON Object

- **case** (*string*) – A case when the evidence is found.
- **data** (*string*) – An evidence itself.
- **id** (*integer*) – A unique evidence identifier.

Status Codes

- 201 Created – An evidence.

Response JSON Object

- **case** (*string*) – A case when the evidence is found.
- **data** (*string*) – An evidence itself.
- **id** (*integer*) – A unique evidence identifier.

GET /evidence/{id}
Show Requested Evidence

Queries and returns an evidence with a passed ID.

Parameters

- **id** (*integer*) – A unique evidence identifier to query.

Status Codes

- 200 OK – An evidence.

- 404 Not Found – Evidence not found.

Request Headers

- If-None-Match – Executes a request only if passed ETag doesn't equal to current resource one (i.e. resource has been changed).

Response Headers

- ETag – Entity tag is part of HTTP provided for web cache validation problem, but also used for optimistic concurrency control.

Response JSON Object

- **`case`** (*`string`*) – A case when the evidence is found.
- **`data`** (*`string`*) – An evidence itself.
- **`id`** (*`integer`*) – A unique evidence identifier.

CHAPTER 2

Options

The `openapi` directive supports the following options:

`encoding` Encoding to be used to read an OpenAPI spec. If not passed, Sphinx's source encoding will be used.

`paths` A comma separated list of paths to filter the included OpenAPI spec by. For example:

```
.. openapi:: specs/openapi.yml
   :paths:
      /persons
      /evidence
   :encoding: utf-8
```

Would only render the endpoints at `/persons` and `/evidence`, ignoring all others.

`examples` If passed, both request and response examples will be rendered. Please note, if examples are not provided in a spec, they will be generated by internal logic based on a corresponding schema.

`group` If passed, paths will be grouped by tags. If a path has no tag assigned, it will be grouped in a `default` group.

`format` The format of text in the spec, either `rst` or `markdown`. If not supplied, ReStructured Text is assumed.

`include` A line separated list of regular expressions to filter the included openapi spec by. For example:

```
.. openapi:: specs/openapi.yml
   :include:
      /evid.*
   :encoding: utf-8
```

Would render the endpoints at `/evidence` and `/evidence/{pk}`

`exclude` A line separated list of regular expressions to filter the included openapi spec by (excluding matches). For example:

```
.. openapi:: specs/openapi.yml
   :exclude:
```

(continues on next page)

(continued from previous page)

```
    /evidence/{pk}
:encoding: utf-8
```

Would render `/persons` and `/evidence` endpoints, but not `/evidence/{pk}` endpoints

`exclude`, `include` and `paths` can also be used together (`exclude` taking precedence over `include` and `paths`)

HTTP Routing Table

/evidence

`GET /evidence`, 6
`GET /evidence/{id}`, 6
`POST /evidence`, 6

/persons

`GET /persons`, 5