

f1557e8f62 ▾

...

ceph / src / pybind / mgr / dashboard / controllers / docs.py / <> Jump to ▾

Aashish Sharma mgr/dashboard: Use secure cookies to store JWT Token ... ✓

🕒 History

👤 10 contributors 

📄 499 lines (441 sloc) | 18.7 KB ...

```
1  # -*- coding: utf-8 -*-
2  from __future__ import absolute_import
3
4  import logging
5  from typing import Any, Dict, List, Union
6
7  import cherrypy
8
9  from .. import DEFAULT_VERSION, mgr
10 from ..api.doc import Schema, SchemaInput, SchemaType
11 from . import ENDPOINT_MAP, BaseController, Controller, Endpoint, allow_empty_body
12
13 NO_DESCRIPTION_AVAILABLE = "No description available"
14
15 logger = logging.getLogger('controllers.docs')
16
17
18 @Controller('/docs', secure=False)
19 class Docs(BaseController):
20
21     @classmethod
22     def _gen_tags(cls, all_endpoints):
23         """ Generates a list of all tags and corresponding descriptions. """
24         # Scenarios to consider:
25         # * Intentionally make up a new tag name at controller => New tag name displayed.
26         # * Misspell or make up a new tag name at endpoint => Neither tag or endpoint displayed.
27         # * Misspell tag name at controller (when referring to another controller) =>
28         # * Tag displayed but no endpoints assigned
29         # * Description for a tag added at multiple locations => Only one description displayed.
30         list_of_ctrl = set()
31         for endpoints in ENDPOINT_MAP.values():
32             for endpoint in endpoints:
33                 if endpoint.is_api or all_endpoints:
34                     list_of_ctrl.add(endpoint.ctrl)
35
36         tag_map: Dict[str, str] = {}
37         for ctrl in list_of_ctrl:
38             tag_name = ctrl.__name__
39             tag_descr = ""
40             if hasattr(ctrl, 'doc_info'):
41                 if ctrl.doc_info['tag']:
42                     tag_name = ctrl.doc_info['tag']
43                     tag_descr = ctrl.doc_info['tag_descr']
44             if tag_name not in tag_map or not tag_map[tag_name]:
45                 tag_map[tag_name] = tag_descr
46
47         tags = [{'name': k, 'description': v if v else NO_DESCRIPTION_AVAILABLE}
48                 for k, v in tag_map.items()]
49         tags.sort(key=lambda e: e['name'])
50         return tags
51
52     @classmethod
53     def _get_tag(cls, endpoint):
54         """ Returns the name of a tag to assign to a path. """
55         ctrl = endpoint.ctrl
56         func = endpoint.func
57         tag = ctrl.__name__
58         if hasattr(func, 'doc_info') and func.doc_info['tag']:
59             tag = func.doc_info['tag']
60         elif hasattr(ctrl, 'doc_info') and ctrl.doc_info['tag']:
61             tag = ctrl.doc_info['tag']
62         return tag
63
64     @classmethod
65     def _gen_type(cls, param):
66         # pylint: disable=too-many-return-statements
67         """
68         Generates the type of parameter based on its name and default value,
69         using very simple heuristics.
70         Used if type is not explicitly defined.
71         """
72         param_name = param['name']
73         def_value = param['default'] if 'default' in param else None
74         if param_name.startswith("is_"):
75             return str(SchemaType.BOOLEAN)
76         if "size" in param_name:
77             return str(SchemaType.INTEGER)
78         if "count" in param_name:
```

```

79         return str(SchemaType.INTEGER)
80     if "num" in param_name:
81         return str(SchemaType.INTEGER)
82     if isinstance(def_value, bool):
83         return str(SchemaType.BOOLEAN)
84     if isinstance(def_value, int):
85         return str(SchemaType.INTEGER)
86     return str(SchemaType.STRING)
87
88 @classmethod
89 # isinstance doesn't work: input is always <type 'type'>.
90 def _type_to_str(cls, type_as_type):
91     """ Used if type is explicitly defined. """
92     if type_as_type is str:
93         type_as_str = str(SchemaType.STRING)
94     elif type_as_type is int:
95         type_as_str = str(SchemaType.INTEGER)
96     elif type_as_type is bool:
97         type_as_str = str(SchemaType.BOOLEAN)
98     elif type_as_type is list or type_as_type is tuple:
99         type_as_str = str(SchemaType.ARRAY)
100     elif type_as_type is float:
101         type_as_str = str(SchemaType.NUMBER)
102     else:
103         type_as_str = str(SchemaType.OBJECT)
104     return type_as_str
105
106 @classmethod
107 def _add_param_info(cls, parameters, p_info):
108     # Cases to consider:
109     # * Parameter name (if not nested) misspelt in decorator => parameter not displayed
110     # * Sometimes a parameter is used for several endpoints (e.g. fs_id in CephFS).
111     # * Currently, there is no possibility of reuse. Should there be?
112     # * But what if there are two parameters with same name but different functionality?
113     """
114     Adds explicitly described information for parameters of an endpoint.
115
116     There are two cases:
117     * Either the parameter in p_info corresponds to an endpoint parameter. Implicit information
118     has higher priority, so only information that doesn't already exist is added.
119     * Or the parameter in p_info describes a nested parameter inside an endpoint parameter.
120     In that case there is no implicit information at all so all explicitly described info needs
121     to be added.
122     """
123     for p in p_info:
124         if not p['nested']:
125             for parameter in parameters:
126                 if p['name'] == parameter['name']:
127                     parameter['type'] = p['type']
128                     parameter['description'] = p['description']
129                     if 'nested_params' in p:
130                         parameter['nested_params'] = cls._add_param_info([], p['nested_params'])
131             else:
132                 nested_p = {
133                     'name': p['name'],
134                     'type': p['type'],
135                     'description': p['description'],
136                     'required': p['required'],
137                 }
138                 if 'default' in p:
139                     nested_p['default'] = p['default']
140                 if 'nested_params' in p:
141                     nested_p['nested_params'] = cls._add_param_info([], p['nested_params'])
142                 parameters.append(nested_p)
143
144     return parameters
145
146 @classmethod
147 def _gen_schema_for_content(cls, params: List[Any]) -> Dict[str, Any]:
148     """
149     Generates information to the content-object in OpenAPI Spec.
150     Used to for request body and responses.
151     """
152     required_params = []
153     properties = {}
154     schema_type = SchemaType.OBJECT
155     if isinstance(params, SchemaInput):
156         schema_type = params.type
157         params = params.params
158
159     for param in params:
160         if param['required']:
161             required_params.append(param['name'])
162
163     props = {}
164     if 'type' in param:
165         props['type'] = cls._type_to_str(param['type'])
166         if 'nested_params' in param:
167             if props['type'] == str(SchemaType.ARRAY): # dict in array
168                 props['items'] = cls._gen_schema_for_content(param['nested_params'])
169             else: # dict in dict
170                 props = cls._gen_schema_for_content(param['nested_params'])
171         elif props['type'] == str(SchemaType.OBJECT): # e.g. [int]
172             props['type'] = str(SchemaType.ARRAY)
173             props['items'] = {'type': cls._type_to_str(param['type'])[0]}
174         else:
175             props['type'] = cls._gen_type(param)
176         if 'description' in param:

```

```

177         props['description'] = param['description']
178     if 'default' in param:
179         props['default'] = param['default']
180     properties[param['name']] = props
181
182     schema = Schema(schema_type=schema_type, properties=properties,
183                     required=required_params)
184
185     return schema.as_dict()
186
187 @classmethod
188 def _gen_responses(cls, method, resp_object=None):
189     resp: Dict[str, Dict[str, Union[str, Any]]] = {
190         '400': {
191             "description": "Operation exception. Please check the "
192                             "response body for details."
193         },
194         '401': {
195             "description": "Unauthenticated access. Please login first."
196         },
197         '403': {
198             "description": "Unauthorized access. Please check your "
199                             "permissions."
200         },
201         '500': {
202             "description": "Unexpected error. Please check the "
203                             "response body for the stack trace."
204         }
205     }
206     if method.lower() == 'get':
207         resp['200'] = {'description': "OK",
208                        'content': {'application/vnd.ceph.api.v{}+json'.format(DEFAULT_VERSION):
209                                   {'type': 'object'}}}
210     if method.lower() == 'post':
211         resp['201'] = {'description': "Resource created.",
212                        'content': {'application/vnd.ceph.api.v{}+json'.format(DEFAULT_VERSION):
213                                   {'type': 'object'}}}
214     if method.lower() == 'put':
215         resp['200'] = {'description': "Resource updated.",
216                        'content': {'application/vnd.ceph.api.v{}+json'.format(DEFAULT_VERSION):
217                                   {'type': 'object'}}}
218     if method.lower() == 'delete':
219         resp['204'] = {'description': "Resource deleted.",
220                        'content': {'application/vnd.ceph.api.v{}+json'.format(DEFAULT_VERSION):
221                                   {'type': 'object'}}}
222     if method.lower() in ['post', 'put', 'delete']:
223         resp['202'] = {'description': "Operation is still executing."
224                                " Please check the task queue.",
225                        'content': {'application/vnd.ceph.api.v{}+json'.format(DEFAULT_VERSION):
226                                   {'type': 'object'}}}
227
228     if resp_object:
229         for status_code, response_body in resp_object.items():
230             if status_code in resp:
231                 resp[status_code].update({
232                     'content': {
233                         'application/vnd.ceph.api.v{}+json'.format(DEFAULT_VERSION): {
234                             'schema': cls._gen_schema_for_content(response_body)}}})
235
236     return resp
237
238 @classmethod
239 def _gen_params(cls, params, location):
240     parameters = []
241     for param in params:
242         if 'type' in param:
243             _type = cls._type_to_str(param['type'])
244         else:
245             _type = cls._gen_type(param)
246         res = {
247             'name': param['name'],
248             'in': location,
249             'schema': {
250                 'type': _type
251             },
252         }
253         if param.get('description'):
254             res['description'] = param['description']
255         if param.get('required'):
256             res['required'] = True
257         elif param.get('default') is None:
258             res['allowEmptyValue'] = True
259         else:
260             res['default'] = param['default']
261         parameters.append(res)
262
263     return parameters
264
265 @classmethod
266 def _gen_paths(cls, all_endpoints):
267     # pylint: disable=R0912
268     method_order = ['get', 'post', 'put', 'delete']
269     paths = {}
270     for path, endpoints in sorted(list(ENDPOINT_MAP.items()),
271                                   key=lambda p: p[0]):
272         methods = {}
273         skip = False
274

```

```

275     endpoint_list = sorted(endpoints, key=lambda e:
276                             method_order.index(e.method.lower()))
277     for endpoint in endpoint_list:
278         if not endpoint.is_api and not all_endpoints:
279             skip = True
280             break
281
282     method = endpoint.method
283     func = endpoint.func
284
285     summary = ''
286     resp = {}
287     p_info = []
288     if hasattr(func, 'doc_info'):
289         if func.doc_info['summary']:
290             summary = func.doc_info['summary']
291         resp = func.doc_info['response']
292         p_info = func.doc_info['parameters']
293     params = []
294     if endpoint.path_params:
295         params.extend(
296             cls._gen_params(
297                 cls._add_param_info(endpoint.path_params, p_info, 'path'))
298     if endpoint.query_params:
299         params.extend(
300             cls._gen_params(
301                 cls._add_param_info(endpoint.query_params, p_info, 'query'))
302
303     methods[method.lower()] = {
304         'tags': [cls._get_tag(endpoint)],
305         'description': func.__doc__,
306         'parameters': params,
307         'responses': cls._gen_responses(method, resp)
308     }
309     if summary:
310         methods[method.lower()][summary] = summary
311
312     if method.lower() in ['post', 'put']:
313         if endpoint.body_params:
314             body_params = cls._add_param_info(endpoint.body_params, p_info)
315             methods[method.lower()][requestBody] = {
316                 'content': {
317                     'application/json': {
318                         'schema': cls._gen_schema_for_content(body_params)}}}
319
320     if endpoint.query_params:
321         query_params = cls._add_param_info(endpoint.query_params, p_info)
322         methods[method.lower()][requestBody] = {
323             'content': {
324                 'application/json': {
325                     'schema': cls._gen_schema_for_content(query_params)}}}
326
327     if endpoint.is_secure:
328         methods[method.lower()][security] = [{'jwt': []}]
329
330     if not skip:
331         paths[path] = methods
332
333     return paths
334
335 @classmethod
336 def _gen_spec(cls, all_endpoints=False, base_url="", offline=False):
337     if all_endpoints:
338         base_url = ""
339
340     host = cherrypy.request.base.split('://', 1)[1] if not offline else 'example.com'
341     logger.debug("Host: %s", host)
342
343     paths = cls._gen_paths(all_endpoints)
344
345     if not base_url:
346         base_url = "/"
347
348     scheme = 'https' if offline or mgr.get_localized_module_option('ssl') else 'http'
349
350     spec = {
351         'openapi': "3.0.0",
352         'info': {
353             'description': "This is the official Ceph REST API",
354             'version': "v1",
355             'title': "Ceph REST API"
356         },
357         'host': host,
358         'basePath': base_url,
359         'servers': [{url: "{}{}".format(
360             cherrypy.request.base if not offline else '',
361             base_url)}],
362         'tags': cls._gen_tags(all_endpoints),
363         'schemes': [scheme],
364         'paths': paths,
365         'components': {
366             'securitySchemes': {
367                 'jwt': {
368                     'type': 'http',
369                     'scheme': 'bearer',
370                     'bearerFormat': 'JWT'
371                 }
372             }
373     }

```

```

373     }
374 }
375
376 return spec
377
378 @Endpoint(path="api.json", version=None)
379 def api_json(self):
380     return self._gen_spec(False, "/")
381
382 @Endpoint(path="api-all.json", version=None)
383 def api_all_json(self):
384     return self._gen_spec(True, "/")
385
386 def _swagger_ui_page(self, all_endpoints=False, token=None):
387     base = cherrypy.request.base
388     if all_endpoints:
389         spec_url = "{}docs/api-all.json".format(base)
390     else:
391         spec_url = "{}docs/api.json".format(base)
392
393     auth_header = cherrypy.request.headers.get('authorization')
394     auth_cookie = cherrypy.request.cookie['token']
395     jwt_token = ""
396     if auth_cookie is not None:
397         jwt_token = auth_cookie.value
398     elif auth_header is not None:
399         scheme, params = auth_header.split(' ', 1)
400         if scheme.lower() == 'bearer':
401             jwt_token = params
402     else:
403         if token is not None:
404             jwt_token = token
405
406     api_key_callback = """onComplete: () => {{
407         ui.preauthorizeApiKey('jwt', '{}');
408     }}
409     """.format(jwt_token)
410
411     page = """
412 <!DOCTYPE html>
413 <html>
414 <head>
415     <meta charset="UTF-8">
416     <meta name="referrer" content="no-referrer" />
417     <link rel="stylesheet" type="text/css"
418         href="/swagger-ui.css" >
419     <style>
420         html
421         {{
422             box-sizing: border-box;
423             overflow: -moz-scrollbars-vertical;
424             overflow-y: scroll;
425         }}
426         *,
427         *:before,
428         *:after
429         {{
430             box-sizing: inherit;
431         }}
432         body {{
433             margin:0;
434             background: #fafafa;
435         }}
436     </style>
437 </head>
438 <body>
439 <div id="swagger-ui"></div>
440 <script src="/swagger-ui-bundle.js">
441 </script>
442 <script>
443     window.onload = function() {{
444         const ui = SwaggerUIBundle({{
445             url: '{}',
446             dom_id: '#swagger-ui',
447             presets: [
448                 SwaggerUIBundle.presets.apis
449             ],
450             layout: "BaseLayout"
451         }})
452         window.ui = ui
453     }}
454 </script>
455 </body>
456 </html>
457     """.format(spec_url, api_key_callback)
458
459     return page
460
461
462 @Endpoint(json_response=False, version=None)
463 def __call__(self, all_endpoints=False):
464     return self._swagger_ui_page(all_endpoints)
465
466 @Endpoint('POST', path="/", json_response=False,
467     query_params="{all_endpoints}", version=None)
468 @allow_empty_body
469 def _with_token(self, token, all_endpoints=False):
470     return self._swagger_ui_page(all_endpoints, token)

```

```
471
472
473 if __name__ == "__main__":
474     import sys
475
476     import yaml
477
478     from . import generate_routes
479
480     def fix_null_descr(obj):
481         """
482         A hot fix for errors caused by null description values when generating
483         static documentation: better fix would be default values in source
484         to be 'None' strings: however, decorator changes didn't resolve
485         """
486         return {k: fix_null_descr(v) for k, v in obj.items() if v is not None} \
487             if isinstance(obj, dict) else obj
488
489     generate_routes("/api")
490     try:
491         with open(sys.argv[1], 'w') as f:
492             # pylint: disable=protected-access
493             yaml.dump(
494                 fix_null_descr(Docs._gen_spec(all_endpoints=False, base_url="/", offline=True)),
495                 f)
496     except IndexError:
497         sys.exit("Output file name missing; correct syntax is: `cmd <file.yml>`")
498     except IsADirectoryError:
499         sys.exit("Specified output is a directory; correct syntax is: `cmd <file.yml>`")
```