
drf-yasg Documentation

Release 1.3.1

Cristi V.

Jan 24, 2018

Table of contents:

1	drf-yasg - Yet another Swagger generator	1
1.1	Features	1
1.2	Table of contents	2
1.3	Usage	4
1.4	Background	8
1.5	Third-party integrations	8
2	Serving the schema	9
2.1	get_schema_view and the SchemaView class	9
2.2	Renderers and codecs	9
2.3	Management command	9
3	Functional overview	11
3.1	OpenAPI specification overview	11
3.2	Default behavior	12
3.3	A note on limitations	14
4	Describing authentication schemes	15
4.1	Security definitions	15
4.2	Security requirements	15
5	Custom schema generation	17
5.1	Excluding endpoints	17
5.2	The @swagger_auto_schema decorator	17
5.3	Serializer Meta nested class	19
5.4	Subclassing and extending	20
6	Customizing the web UI	23
7	Settings	25
7.1	SWAGGER_SETTINGS	25
7.2	REDOC_SETTINGS	29
8	Contributing	31
8.1	Issues	31
8.2	Pull requests	31
8.3	Maintainer's notes	32

9 License	35
9.1 BSD 3-Clause License	35
10 Changelog	37
10.1 1.3.1	37
10.2 1.3.0	37
10.3 1.2.2	37
10.4 1.2.1	37
10.5 1.2.0	38
10.6 1.1.3	38
10.7 1.1.2	38
10.8 1.1.1	38
10.9 1.1.0	38
10.10 1.0.6	38
10.11 1.0.5	39
10.12 1.0.4	39
10.13 1.0.3	39
10.14 1.0.2	39
11 Source code documentation	41
11.1 drf_yasg package	41
Python Module Index	69

drf-yasg - Yet another Swagger generator

Generate **real** Swagger/OpenAPI 2.0 specifications from a Django Rest Framework API.

Compatible with

- **Django Rest Framework:** 3.7.7
- **Django:** 1.11.x, 2.0.x
- **Python:** 2.7, 3.4, 3.5, 3.6

Resources:

- **Source:** <https://github.com/axnsan12/drf-yasg/>
- **Documentation:** <https://drf-yasg.readthedocs.io/>
- **Changelog:** <https://drf-yasg.readthedocs.io/en/stable/changelog.html>
- **Live demo:** <https://drf-yasg-demo.herokuapp.com/>

1.1 Features

- full support for nested Serializers and Schemas
- response schemas and descriptions
- model definitions compatible with codegen tools
- customization hooks at all points in the spec generation process
- JSON and YAML format for spec
- bundles latest version of `swagger-ui` and `redoc` for viewing the generated documentation
- schema view is cacheable out of the box

- generated Swagger schema can be automatically validated by [swagger-spec-validator](#) or [flex](#)
- supports Django REST Framework API versioning with [URLPathVersioning](#) and [NamespaceVersioning](#); other DRF or custom versioning schemes are not currently supported

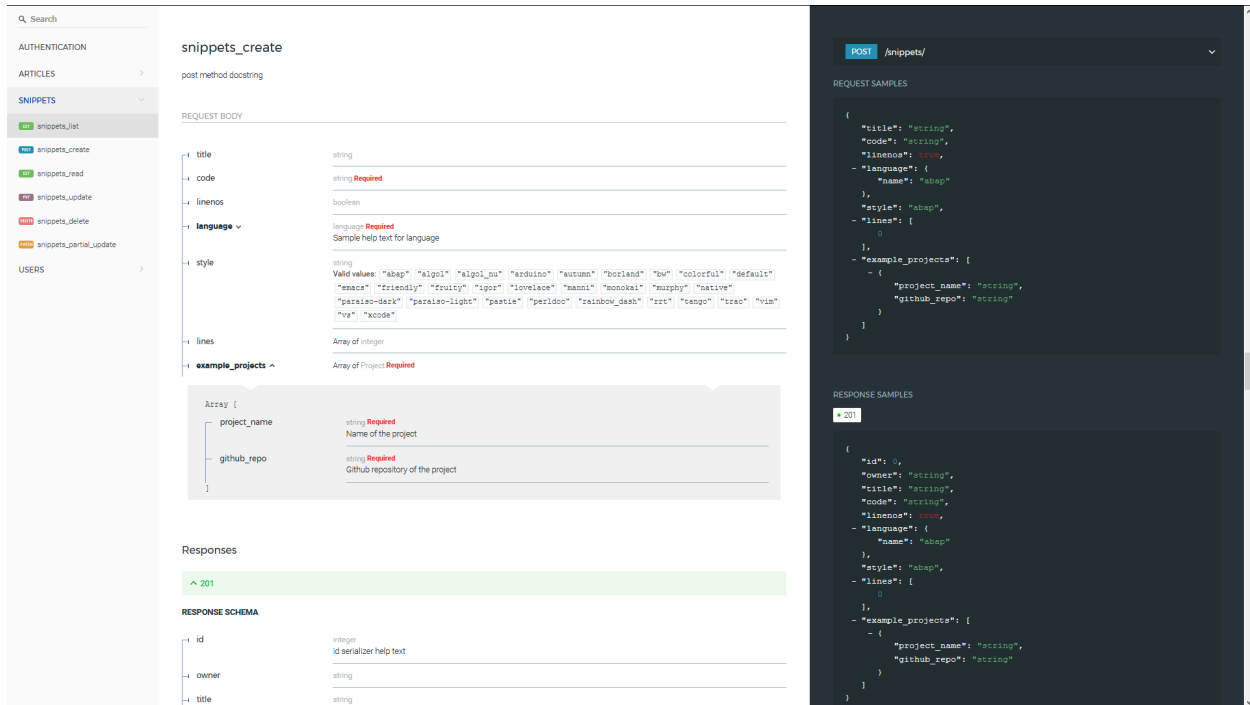


Fig. 1.1: Fully nested request and response schemas.

1.2 Table of contents

Contents

- *drf-yasg - Yet another Swagger generator*
 - *Features*
 - *Table of contents*
 - *Usage*
 - * *0. Installation*
 - * *1. Quickstart*
 - * *2. Configuration*
 - *a. get_schema_view parameters*
 - *b. SchemaView options*
 - *c. SWAGGER_SETTINGS and REDOC_SETTINGS*
 - * *3. Caching*

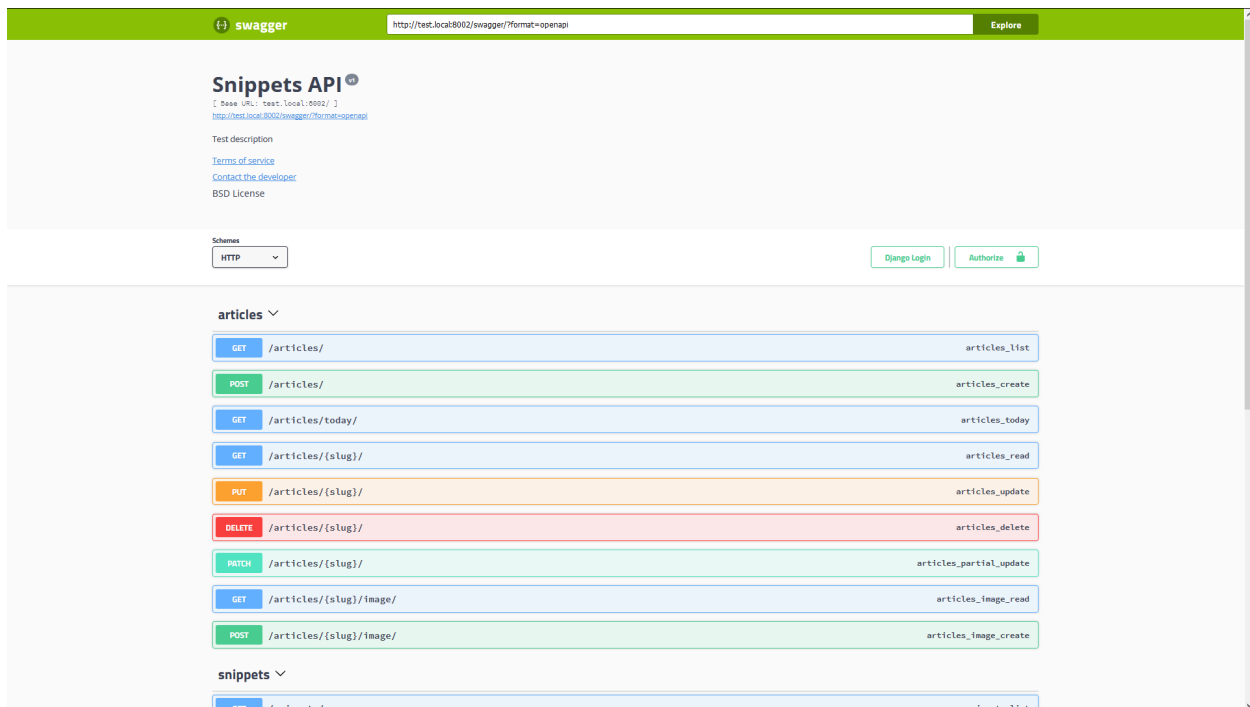


Fig. 1.2: Choose between redoc and swagger-ui.



Fig. 1.3: Real Model definitions.

- * 4. *Validation*
 - *swagger-ui validation badge*
 - *Using swagger-cli*
 - *Manually on editor.swagger.io*
- * 5. *Code generation*
- * 6. *Example project*
- *Background*
 - * *Swagger in Django Rest Framework*
 - * *Other libraries*
- *Third-party integrations*
 - * *djangoestframework-camel-case*

1.3 Usage

1.3.1 0. Installation

The preferred installation method is directly from pypi:

```
pip install drf-yasg
```

Additionally, if you want to use the built-in validation mechanisms (see [4. Validation](#)), you need to install some extra requirements:

```
pip install drf-yasg[validation]
```

1.3.2 1. Quickstart

In `settings.py`:

```
INSTALLED_APPS = [  
    ...  
    'drf_yasg',  
    ...  
]
```

In `urls.py`:

```
...  
from drf_yasg.views import get_schema_view  
from drf_yasg import openapi  
...  
  
schema_view = get_schema_view(  
    openapi.Info(  
        title="Snippets API",
```



```

        default_version='v1',
        description="Test description",
        terms_of_service="https://www.google.com/policies/terms/",
        contact=openapi.Contact(email="contact@snippets.local"),
        license=openapi.License(name="BSD License"),
    ),
    validators=['flex', 'ssv'],
    public=True,
    permission_classes=(permissions.AllowAny,),
)

urlpatterns = [
    url(r'^swagger(?:P<format>.json|.yaml)$', schema_view.without_ui(cache_
↪ timeout=None), name='schema-json'),
    url(r'^swagger/$', schema_view.with_ui('swagger', cache_timeout=None), name=
↪ 'schema-swagger-ui'),
    url(r'^redoc/$', schema_view.with_ui('redoc', cache_timeout=None), name='schema-
↪ redoc'),
    ...
]

```

This exposes 4 cached, validated and publicly available endpoints:

- A JSON view of your API specification at `/swagger.json`
- A YAML view of your API specification at `/swagger.yaml`
- A swagger-ui view of your API specification at `/swagger/`
- A ReDoc view of your API specification at `/redoc/`

1.3.3 2. Configuration

a. `get_schema_view` parameters

- `info` - Swagger API Info object; if omitted, defaults to `DEFAULT_INFO`
- `url` - API base url; if left blank will be deduced from the location the view is served at
- `patterns` - passed to `SchemaGenerator`
- `urlconf` - passed to `SchemaGenerator`
- `public` - if `False`, includes only endpoints the current user has access to
- `validators` - a list of validator names to apply on the generated schema; allowed values are `flex`, `ssv`
- `generator_class` - schema generator class to use; should be a subclass of `OpenAPISchemaGenerator`
- `authentication_classes` - authentication classes for the schema view itself
- `permission_classes` - permission classes for the schema view itself

b. `SchemaView` options

- `SchemaView.with_ui(renderer, cache_timeout, cache_kwargs)` - get a view instance using the specified UI renderer; one of `swagger`, `redoc`
- `SchemaView.without_ui(cache_timeout, cache_kwargs)` - get a view instance with no UI renderer; same as `as_cached_view` with no kwargs

- `SchemaView.as_cached_view(cache_timeout, cache_kwargs, **initkwargs)` - same as `as_view`, but with optional caching
- you can, of course, call `as_view` as usual

All of the first 3 methods take two optional arguments, `cache_timeout` and `cache_kwargs`; if present, these are passed on to Django's `cached_page` decorator in order to enable caching on the resulting view. See [3. Caching](#).

C. SWAGGER_SETTINGS and REDOC_SETTINGS

Additionally, you can include some more settings in your `settings.py` file. See <https://drf-yasg.readthedocs.io/en/stable/settings.html> for details.

1.3.4 3. Caching

Since the schema does not usually change during the lifetime of the django process, there is out of the box support for caching the schema view in-memory, with some sane defaults:

- caching is enabled by the `cache_page` decorator, using the default Django cache backend, can be changed using the `cache_kwargs` argument
- HTTP caching of the response is blocked to avoid confusing situations caused by being shown stale schemas
- the cached schema varies on the `Cookie` and `Authorization` HTTP headers to enable filtering of visible endpoints according to the authentication credentials of each user; note that this means that every user accessing the schema will have a separate schema cached in memory.

1.3.5 4. Validation

Given the numerous methods to manually customize the generated schema, it makes sense to validate the result to ensure it still conforms to OpenAPI 2.0. To this end, validation is provided at the generation point using python swagger libraries, and can be activated by passing `validators=['flex', 'ssv']` to `get_schema_view`; if the generated schema is not valid, a `SwaggerValidationError` is raised by the handling codec.

Warning: This internal validation can slow down your server. Caching can mitigate the speed impact of validation.

The provided validation will catch syntactic errors, but more subtle violations of the spec might slip by them. To ensure compatibility with code generation tools, it is recommended to also employ one or more of the following methods:

swagger-ui validation badge

Online

If your schema is publicly accessible, *swagger-ui* will automatically validate it against the official swagger online validator and display the result in the bottom-right validation badge.

Offline

If your schema is not accessible from the internet, you can run a local copy of *swagger-validator* and set the `VALIDATOR_URL` accordingly:

```
SWAGGER_SETTINGS = {
    ...
    'VALIDATOR_URL': 'http://localhost:8189',
    ...
}
```

```
$ docker run --name swagger-validator -d -p 8189:8080 --add-host test.local:10.0.75.1
↪ swaggerapi/swagger-validator
84dabd52ba967c32ae6b660934fa6a429ca6bc9e594d56e822a858b57039c8a2
$ curl http://localhost:8189/debug?url=http://test.local:8002/swagger/?format=openapi
{}
```

Using `swagger-cli`

<https://www.npmjs.com/package/swagger-cli>

```
$ npm install -g swagger-cli
[...]
$ swagger-cli validate http://test.local:8002/swagger.yaml
http://test.local:8002/swagger.yaml is valid
```

Manually on editor.swagger.io

Importing the generated spec into <https://editor.swagger.io/> will automatically trigger validation on it. This method is currently the only way to get both syntactic and semantic validation on your specification. The other validators only provide JSON schema-level validation, but miss things like duplicate operation names, improper content types, etc

1.3.6 5. Code generation

You can use the specification outputted by this library together with `swagger-codegen` to generate client code in your language of choice:

```
$ docker run --rm -v ${PWD}:/local swaggerapi/swagger-codegen-cli generate -i /local/
↪ tests/reference.yaml -l javascript -o /local/.codegen/js
```

See the github page linked above for more details.

1.3.7 6. Example project

For additional usage examples, you can take a look at the test project in the `testproj` directory:

```
$ git clone https://github.com/axnsan12/drf-yasg.git
$ cd drf-yasg
$ virtualenv venv
$ source venv/bin/activate
(venv) $ cd testproj
(venv) $ pip install -r requirements.txt
(venv) $ python manage.py migrate
(venv) $ python manage.py shell -c "import createsuperuser"
(venv) $ python manage.py runserver
(venv) $ firefox localhost:8000/swagger/
```

1.4 Background

OpenAPI 2.0/Swagger is a format designed to encode information about a Web API into an easily parsable schema that can then be used for rendering documentation, generating code, etc.

More details are available on swagger.io and on the [OpenAPI 2.0 specification page](#).

From here on, the terms “OpenAPI” and “Swagger” are used interchangeably.

1.4.1 Swagger in Django Rest Framework

Since Django Rest 3.7, there is now [built in support](#) for automatic OpenAPI 2.0 schema generation. However, this generation is based on the [coreapi](#) standard, which for the moment is vastly inferior to OpenAPI in both features and tooling support. In particular, the OpenAPI codec/compatibility layer provided has a few major problems:

- there is no support for documenting response schemas and status codes
- nested schemas do not work properly
- does not handle more complex fields such as `FileField`, `ChoiceField`, ...

In short this makes the generated schema unusable for code generation, and mediocre at best for documentation.

1.4.2 Other libraries

There are currently two decent Swagger schema generators that I could find for `django-rest-framework`:

- `django-rest-swagger`
- `drf-openapi`

Out of the two, `django-rest-swagger` is just a wrapper around DRF 3.7 schema generation with an added UI, and thus presents the same problems. `drf-openapi` is a bit more involved and implements some custom handling for response schemas, but ultimately still falls short in code generation because the responses are plain of lacking support for named schemas.

Both projects are also currently unmaintained.

1.5 Third-party integrations

1.5.1 `django-rest-framework-camel-case`

Integration with `django-rest-framework-camel-case` is provided out of the box - if you have `django-rest-framework-camel-case` installed and your `APIView` uses `CamelCaseJSONParser` or `CamelCaseJSONRenderer`, all property names will be converted to *camelCase* by default.

Serving the schema

2.1 `get_schema_view` and the `SchemaView` class

The `get_schema_view()` function and the `SchemaView` class it returns (click links for documentation) are intended to cover the majority of use cases one might want to configure. The class returned by `get_schema_view()` can be used to obtain view instances via `SchemaView.with_ui()`, `SchemaView.without_ui()` and `SchemaView.as_cached_view()` - see *1. Quickstart* in the README for a usage example.

You can also subclass `SchemaView` by extending the return value of `get_schema_view()`, e.g.:

```
SchemaView = get_schema_view(info, ...)

class CustomSchemaView(SchemaView):
    generator_class = CustomSchemaGenerator
    renderer_classes = (CustomRenderer1, CustomRenderer2,)
```

2.2 Renderers and codecs

If you need to modify how your Swagger spec is presented in views, you might want to override one of the renderers in `renderers` or one of the codecs in `codecs`. The codec is the last stage where the Swagger object arrives before being transformed into bytes, while the renderer is the stage responsible for tying together the codec and the view.

You can use your custom renderer classes as kwargs to `SchemaView.as_cached_view()` or by subclassing `SchemaView`.

2.3 Management command

If you only need a swagger spec file in YAML or JSON format, you can use the `generate_swagger` management command to get it without having to start the web server:

```
$ python manage.py generate_swagger swagger.json
```

See the command help for more advanced options:

```
$ python manage.py generate_swagger --help
usage: manage.py generate_swagger [-h] [--version] [-v {0,1,2,3}]
    ... more options ...
```

Note: The `DEFAULT_INFO` setting must be defined when using the `generate_swagger` command. For example, the `README quickstart` code could be modified as such:

In `settings.py`:

```
SWAGGER_SETTINGS = {
    'DEFAULT_INFO': 'import.path.to.urls.api_info',
}
```

In `urls.py`:

```
api_info = openapi.Info(
    title="Snippets API",
    ... other arguments ...
)

schema_view = get_schema_view(
    # the info argument is no longer needed here as it will be picked up from DEFAULT_
    ↪ INFO
    ... other arguments ...
)
```

3.1 OpenAPI specification overview

This library generates OpenAPI 2.0 documents. The authoritative specification for this document's structure will always be the official documentation over at swagger.io and the [OpenAPI 2.0 specification page](#).

Because the above specifications are a bit heavy and convoluted, here is a general overview of how the specification is structured, starting from the root `Swagger` object.

- **Swagger object**
 - `info`, `schemes`, `securityDefinitions` and other informative attributes
 - **paths: Paths object** A list of all the paths in the API in the form of a mapping
 - * **{path}: PathItem** - each *PathItem* has multiple operations keyed by method
 - **{http_method}: Operation** Each operation is thus uniquely identified by its (path, http_method) combination, e.g. GET /articles/, POST /articles/, etc.
 - `parameters: [Parameter]` - and a list of path parameters
 - **definitions: named Models** A list of all the named models in the API in the form of a mapping
 - * `{modelName}: Schema`
- **Operation contains the following information about each operation:**
 - **parameters: [Parameter]** A list of all the *query*, *header* and *form* parameters accepted by the operation.
 - * there can also be **at most one** body parameter whose structure is represented by a *Schema* or a reference to one (*SchemaRef*)
 - **responses: Responses** A list of all the possible responses the operation is expected to return. Each response can optionally have a *Schema* which describes the structure of its body.
 - * `{status_code}: Response` - mapping of status code to response definition

- `operationId` - should be unique across all operations
- `tags` - used to group operations in the listing

It is interesting to note the main differences between *Parameter* and *Schema* objects:

<i>Schema</i>	<i>Parameter</i>
Can nest other Schemas	Cannot nest other Parameters Can only nest a Schema if the parameter is <code>in: body</code>
Cannot describe file uploads - <code>file</code> is not permitted as a value for <code>type</code>	Can describe file uploads via <code>type = file</code> , but only as part of a form <i>Operation</i> ¹
Can be used in <i>Responses</i>	Cannot be used in <i>Responses</i>
Cannot be used in form <i>Operations</i> ¹	Can be used in form <i>Operations</i> ¹
Can only describe request or response bodies	Can describe query, form, header or path parameters

3.2 Default behavior

This section describes where information is sourced from when using the default generation process.

- *Paths* are generated by exploring the patterns registered in your default `urlpatterns`, or the patterns and `urlpatterns` you specified when constructing *OpenAPISchemaGenerator*; only views inheriting from Django Rest Framework's `APIView` are looked at, all other views are ignored
- path *Parameters* are generated by looking in the URL pattern for any template parameters; attempts are made to guess their type from the views `queryset` and `lookup_field`, if applicable. You can override path parameters via `manual_parameters` in *@swagger_auto_schema*.
- query *Parameters* - i.e. parameters specified in the URL as `/path/?query1=value&query2=value` - are generated from your view's `filter_backends` and `paginator`, if any are declared. Additional parameters can be specified via the `query_serializer` and `manual_parameters` arguments of *@swagger_auto_schema*
- The request body is only generated for the HTTP POST, PUT and PATCH methods, and is sourced from the view's `serializer_class`. You can also override the request body using the `request_body` argument of *@swagger_auto_schema*.
 - if the view represents a form request (that is, all its parsers are of the `multipart/form-data` or `application/x-www-form-urlencoded` media types), the request body will be output as form *Parameters*
 - if it is not a form request, the request body will be output as a single *body Parameter* wrapped around a *Schema*
- header *Parameters* are supported by the OpenAPI specification but are never generated by this library; you can still add them using `manual_parameters`.
- *Responses* are generated as follows:
 - if `responses` is provided to *@swagger_auto_schema* and contains at least one success status code (i.e. any `2xx` status code), no automatic response is generated and the given response is used as described in the *@swagger_auto_schema documentation*

¹ a form *Operation* is an *Operation* that consumes `multipart/form-data` or `application/x-www-form-urlencoded` content

- a form *Operation* cannot have `body` parameters
- a non-form operation cannot have `form` parameters

- otherwise, an attempt is made to generate a default response:
 - * the success status code is assumed to be 204` for ``DELETE requests, 201 for POST requests, and 200 for all other request methods
 - * if the view has a request body, the same `Serializer` or `Schema` as in the request body is used in generating the `Response` schema; this is inline with the default `GenericAPIView` and `GenericViewSet` behavior
 - * if the view has no request body, its `serializer_class` is used to generate the `Response` schema
 - * if the view is a list view (as defined by `is_list_view()`), the response schema is wrapped in an array
 - * if the view is also paginated, the response schema is then wrapped in the appropriate paging response structure
 - * the description of the response is left blank
- `Response` headers are supported by the OpenAPI specification but not currently supported by this library; you can still add them manually by providing an `appropriately structured dictionary` to the `headers` property of a `Response` object
- `descriptions` for `Operations`, `Parameters` and `Schemas` are picked up from docstrings and `help_text` attributes in the same manner as the `default DRF SchemaGenerator`
- The base URL for the API consists of three values - the `host`, `schemes` and `basePath` attributes
- The host name and scheme are determined, in descending order of priority:
 - from the `url` argument passed to `get_schema_view()` (more specifically, to the underlying `OpenAPISchemaGenerator`)
 - from the `DEFAULT_API_URL` setting
 - inferred from the request made to the schema endpoint

For example, an url of `https://www.example.com:8080/some/path` will populate the `host` and `schemes` attributes with `www.example.com:8080` and `['https']`, respectively. The path component will be ignored.

- The base path is determined as the concatenation of two variables:
 1. the `SCRIPT_NAME` wsgi environment variable; this is set, for example, when serving the site from a sub-path using web server url rewriting

Tip: The Django `FORCE_SCRIPT_NAME` setting can be used to override the `SCRIPT_NAME` or set it when it's missing from the environment.

2. the longest common path prefix of all the urls in your API - see `determine_path_prefix()`
- When using API versioning with `NamespaceVersioning` or `URLPathVersioning`, versioned endpoints that do not match the version used to access the `SchemaView` will be excluded from the endpoint list - for example, `/api/v1.0/endpoint` will be shown when viewing `/api/v1.0/swagger/`, while `/api/v2.0/endpoint` will not
- Other versioning schemes are not presently supported.

3.3 A note on limitations

When schema generation is requested, available endpoints are inspected by enumeration all the routes registered in Django's `urlpatterns`. Each registered view is then artificially instantiated for introspection, and it is this step that brings some limitations to what can be done:

- the `request` the view sees will always be the request made against the schema view endpoint - e.g. `GET /swagger.yaml`
- path parameters will not be filled

This means that you could get surprising results if your `get_serializer` or `get_serializer_class` methods depend on the incoming request, call `get_object` or in general depend on any stateful logic. You can prevent this in a few ways:

- provide a fixed serializer for request and response body introspection using `@swagger_auto_schema`, to prevent `get_serializer` from being called on the view
- *exclude your endpoint from introspection*

Describing authentication schemes

When using the *swagger-ui* frontend, it is possible to interact with the API described by your Swagger document. This interaction might require authentication, which you will have to describe in order to make *swagger-ui* work with it.

4.1 Security definitions

The first step that you have to do is add a *SECURITY_DEFINITIONS* setting to declare all authentication schemes supported by your API.

For example, the definition for a simple API accepting HTTP basic auth and *Authorization* header API tokens would be:

```
SWAGGER_SETTINGS = {
  'SECURITY_DEFINITIONS': {
    'Basic': {
      'type': 'basic'
    },
    'Bearer': {
      'type': 'apiKey',
      'name': 'Authorization',
      'in': 'header'
    }
  }
}
```

4.2 Security requirements

The second step is specifying, for each endpoint, which authentication mechanism can be used for interacting with it. See <https://github.com/OAI/OpenAPI-Specification/blob/master/versions/2.0.md#security-requirement-object> for details.

By default, a top-level *security* that accepts all the declared security definitions is generated. For the example above, that would be `[{'Basic': []}, {'Bearer': []}]`. This can be overridden using the *SECURITY_REQUIREMENTS* setting.

Operation-level overrides can be added using the *security* parameter of `@swagger_auto_schema`.

Custom schema generation

If the default spec generation does not quite match what you were hoping to achieve, `drf-yasg` provides some custom behavior hooks by default.

5.1 Excluding endpoints

You can prevent a view from being included in the Swagger view by setting its class-level `swagger_schema` attribute to `None`, or you can prevent an operation from being included by setting its `auto_schema` override to `none` in `@swagger_auto_schema`:

```
class UserList(APIView):
    swagger_schema = None

    # all methods of the UserList class will be excluded
    ...

# only the GET method will be shown in Swagger
@swagger_auto_schema(method='put', auto_schema=None)
@swagger_auto_schema(methods=['get'], ...)
@api_view(['GET', 'PUT'])
def user_detail(request, pk):
    pass
```

5.2 The `@swagger_auto_schema` decorator

You can use the `@swagger_auto_schema` decorator on view functions to override some properties of the generated `Operation`. For example, in a `ViewSet`,

```
@swagger_auto_schema(operation_description="partial_update description override",
↳ responses={404: 'slug not found'})
def partial_update(self, request, *args, **kwargs):
```

```
"""partial_update method docstring"""
...
```

will override the description of the PATCH `/article/{id}/` operation, and document a 404 response with no body and the given description.

Where you can use the `@swagger_auto_schema` decorator depends on the type of your view:

- for function based `@api_views`, because the same view can handle multiple methods, and thus represent multiple operations, you have to add the decorator multiple times if you want to override different operations:

```
test_param = openapi.Parameter('test', openapi.IN_QUERY, description=
↳ "test manual param", type=openapi.TYPE_BOOLEAN)
user_response = openapi.Response('response description', UserSerializer)

# 'method' can be used to customize a single HTTP method of a view
@swagger_auto_schema(method='get', manual_parameters=[test_param],
↳ responses={200: user_response})
# 'methods' can be used to apply the same modification to multiple
↳ methods
@swagger_auto_schema(methods=['put', 'post'], request_
↳ body=UserSerializer)
@api_view(['GET', 'PUT', 'POST'])
def user_detail(request, pk):
    ...
```

- for class based `APIView`, `GenericAPIView` and non-`ViewSet` derivatives, you have to decorate the respective method of each operation:

```
class UserList(APIView):
    @swagger_auto_schema(responses={200: UserSerializer(many=True)})
    def get(self, request):
        ...

    @swagger_auto_schema(operation_description="description")
    def post(self, request):
        ...
```

- for `ViewSet`, `GenericViewSet`, `ModelViewSet`, because each viewset corresponds to multiple **paths**, you have to decorate the *action methods*, i.e. `list`, `create`, `retrieve`, etc. Additionally, `@list_routes` or `@detail_routes` defined on the viewset, like function based api views, can respond to multiple HTTP methods and thus have multiple operations that must be decorated separately:

```
class ArticleViewSet(viewsets.ModelViewSet):
    # method or 'methods' can be skipped because the list_route only
↳ handles a single method (GET)
    @swagger_auto_schema(operation_description='GET /articles/today/')
    @list_route(methods=['get'])
    def today(self, request):
        ...

    @swagger_auto_schema(method='get', operation_description="GET /
↳ articles/{id}/image/")
    @swagger_auto_schema(method='post', operation_description="POST /
↳ articles/{id}/image/")
    @detail_route(methods=['get', 'post'], parser_
↳ classes=(MultiPartParser,))
    def image(self, request, id=None):
```

```

...

@swagger_auto_schema(operation_description="PUT /articles/{id}/")
def update(self, request, *args, **kwargs):
    ...

@swagger_auto_schema(operation_description="PATCH /articles/{id}/")
def partial_update(self, request, *args, **kwargs):
    ...

```

Tip: If you want to customize the generation of a method you are not implementing yourself, you can use `swagger_auto_schema` in combination with Django's `method_decorator`:

```

@method_decorator(name='list', decorator=swagger_auto_schema(
    operation_description="description from swagger_auto_schema via method_decorator"
))
class ArticleViewSet(viewsets.ModelViewSet):
    ...

```

This allows you to avoid unnecessarily overriding the method.

Tip: You can go even further and directly decorate the result of `as_view`, in the same manner you would override an `@api_view` as described above:

```

decorated_login_view = \
    swagger_auto_schema(
        method='post',
        responses={status.HTTP_200_OK: LoginResponseSerializer}
    )(LoginView.as_view())

urlpatterns = [
    ...
    url(r'^login/$', decorated_login_view, name='login')
]

```

This can allow you to avoid skipping an unnecessary *subclass* altogether.

Warning: However, do note that both of the methods above can lead to unexpected (and maybe surprising) results by replacing/decorating methods on the base class itself.

5.3 Serializer Meta nested class

You can define some per-serializer options by adding a `Meta` class to your serializer, e.g.:

```

class WhateverSerializer(serializer.Serializer):
    ...

    class Meta:
        ... options here ...

```

Currently, the only option you can add here is

- `ref_name` - a string which will be used as the model definition name for this serializer class; setting it to `None` will force the serializer to be generated as an inline model everywhere it is used

5.4 Subclassing and extending

5.4.1 `SwaggerAutoSchema`

For more advanced control you can subclass `SwaggerAutoSchema` - see the documentation page for a list of methods you can override.

You can put your custom subclass to use by setting it on a view method using the `@swagger_auto_schema` decorator described above, by setting it as a class-level attribute named `swagger_schema` on the view class, or *globally via settings*.

For example, to generate all operation IDs as camel case, you could do:

```
from inflection import camelize

class CamelCaseOperationIDAutoSchema(SwaggerAutoSchema):
    def get_operation_id(self, operation_keys):
        operation_id = super(CamelCaseOperationIDAutoSchema, self).get_operation_
        ↪id(operation_keys)
        return camelize(operation_id, uppercase_first_letter=False)

SWAGGER_SETTINGS = {
    'DEFAULT_AUTO_SCHEMA_CLASS': 'path.to.CamelCaseOperationIDAutoSchema',
    ...
}
```

5.4.2 `OpenAPISchemaGenerator`

If you need to control things at a higher level than `Operation` objects (e.g. overall document structure, vendor extensions in metadata) you can also subclass `OpenAPISchemaGenerator` - again, see the documentation page for a list of its methods.

This custom generator can be put to use by setting it as the `generator_class` of a `SchemaView` using `get_schema_view()`.

5.4.3 `Inspector` classes

For customizing behavior related to specific field, serializer, filter or paginator classes you can implement the `FieldInspector`, `SerializerInspector`, `FilterInspector`, `PaginatorInspector` classes and use them with `@swagger_auto_schema` or one of the *related settings*.

A `FilterInspector` that adds a description to all DjangoFilterBackend parameters could be implemented like so:

```
class DjangoFilterDescriptionInspector(CoreAPICompatInspector):
    def get_filter_parameters(self, filter_backend):
        if isinstance(filter_backend, DjangoFilterBackend):
            result = super(DjangoFilterDescriptionInspector, self).get_filter_
            ↪parameters(filter_backend)
```



```

        for param in result:
            if not param.get('description', ''):
                param.description = "Filter the returned list by {field_name}".
↪format(field_name=param.name)

            return result

        return NotHandled

@method_decorator(name='list', decorator=swagger_auto_schema(
    filter_inspectors=[DjangoFilterDescriptionInspector]
))
class ArticleViewSet(viewsets.ModelViewSet):
    filter_backends = (DjangoFilterBackend,)
    filter_fields = ('title',)
    ...

```

A second example, of a *FieldInspector* that removes the title attribute from all generated *Schema* objects:

```

class NoSchemaTitleInspector(FieldInspector):
    def process_result(self, result, method_name, obj, **kwargs):
        # remove the `title` attribute of all Schema objects
        if isinstance(result, openapi.Schema.OR_REF):
            # traverse any references and alter the Schema object in place
            schema = openapi.resolve_ref(result, self.components)
            schema.pop('title', None)

            # no ``return schema`` here, because it would mean we always generate
            # an inline `object` instead of a definition reference

            # return back the same object that we got - i.e. a reference if we got a ↪
↪reference
            return result

class NoTitleAutoSchema(SwaggerAutoSchema):
    field_inspectors = [NoSchemaTitleInspector] + swagger_settings.DEFAULT_FIELD_
↪INSPECTORS

class ArticleViewSet(viewsets.ModelViewSet):
    swagger_schema = NoTitleAutoSchema
    ...

```

Note: A note on references - *Schema* objects are sometimes output by reference (*SchemaRef*); in fact, that is how named models are implemented in OpenAPI:

- in the output swagger document there is a definitions section containing *Schema* objects for all models
- every usage of a model refers to that single *Schema* object - for example, in the ArticleViewSet above, all requests and responses containing an Article model would refer to the same schema definition by a '\$ref': '#/definitions/Article'

This is implemented by only generating **one** *Schema* object for every serializer **class** encountered.

This means that you should generally avoid view or method-specific *FieldInspectors* if you are dealing with references (a.k.a named models), because you can never know which view will be the first to generate the schema for a given serializer.

CHAPTER 6

Customizing the web UI

There is currently no pluggable way of customizing the web UI apart from the settings available in *Swagger UI settings* and *ReDoc UI settings*. If you really need to, you can override one of the `drf-yasg/swagger-ui.html` or `drf-yasg/redoc.html` templates that are used for rendering.

Settings are configurable in `settings.py` by defining `SWAGGER_SETTINGS` or `REDOC_SETTINGS`.

Example:

settings.py

```
SWAGGER_SETTINGS = {
    'SECURITY_DEFINITIONS': {
        'basic': {
            'type': 'basic'
        }
    },
    ...
}

REDOC_SETTINGS = {
    'LAZY_RENDERING': True,
    ...
}
```

The possible settings and their default values are as follows:

7.1 SWAGGER_SETTINGS

7.1.1 Default classes

DEFAULT_AUTO_SCHEMA_CLASS

ViewInspector subclass that will be used by default for generating *Operation* objects when iterating over endpoints. Can be overridden by using the *auto_schema* argument of `@swagger_auto_schema` or by a `swagger_schema` attribute on the view class.

Default: `drf_yasg.inspectors.SwaggerAutoSchema`

DEFAULT_FIELD_INSPECTORS

List of *FieldInspector* subclasses that will be used by default for inspecting serializers and serializer fields. Field inspectors given to `@swagger_auto_schema` will be prepended to this list.

Default: [`'drf_yasg.inspectors.CamelCaseJSONFilter'`, `'drf_yasg.inspectors.ReferencingSerializerInspector'`, `'drf_yasg.inspectors.RelatedFieldInspector'`, `'drf_yasg.inspectors.ChoiceFieldInspector'`, `'drf_yasg.inspectors.FileFieldInspector'`, `'drf_yasg.inspectors.DictFieldInspector'`, `'drf_yasg.inspectors.SimpleFieldInspector'`, `'drf_yasg.inspectors.StringDefaultFieldInspector'`,]

DEFAULT_FILTER_INSPECTORS

List of *FilterInspector* subclasses that will be used by default for inspecting filter backends. Filter inspectors given to `@swagger_auto_schema` will be prepended to this list.

Default: [`'drf_yasg.inspectors.CoreAPICompatInspector'`,]

DEFAULT_PAGINATOR_INSPECTORS

List of *PaginatorInspector* subclasses that will be used by default for inspecting paginators. Paginator inspectors given to `@swagger_auto_schema` will be prepended to this list.

Default: [`'drf_yasg.inspectors.DjangoRestResponsePagination'`, `'drf_yasg.inspectors.CoreAPICompatInspector'`,]

7.1.2 Swagger document attributes

DEFAULT_INFO

An import string to an *openapi.Info* object. This will be used when running the `generate_swagger` management command, or if no `info` argument is passed to `get_schema_view`.

Default: None

DEFAULT_API_URL

A string representing the default API URL. This will be used to populate the `host` and `schemes` attributes of the Swagger document if no API URL is otherwise provided. The Django `FORCE_SCRIPT_NAME` setting can be used for providing an API mount point prefix.

See also: *documentation on base URL construction*

Default: None

7.1.3 Authorization

USE_SESSION_AUTH

Enable/disable Django login as an authentication/authorization mechanism. If True, a login/logout button will be displayed in Swagger UI.

Default: True

LOGIN_URL

URL for the Django Login action when using `USE_SESSION_AUTH`.

Default: `django.conf.settings.LOGIN_URL`

LOGOUT_URL

URL for the Django Logout action when using `USE_SESSION_AUTH`.

Default: `django.conf.settings.LOGOUT_URL`

SECURITY_DEFINITIONS

Swagger security definitions to be included in the specification. See <https://github.com/OAI/OpenAPI-Specification/blob/master/versions/2.0.md#security-definitions-object>.

Default:

```
'basic': {
  'type': 'basic'
}
```

SECURITY_REQUIREMENTS

Global security requirements. If None, all schemes in `SECURITY_DEFINITIONS` are accepted. See <https://github.com/OAI/OpenAPI-Specification/blob/master/versions/2.0.md#securityRequirementObject>.

Default: None

7.1.4 Swagger UI settings

Swagger UI configuration settings. See <https://github.com/swagger-api/swagger-ui/blob/112bca906553a937ac67adc2e500bdeed96d067b/docs/usage/configuration.md#parameters>.

VALIDATOR_URL

URL pointing to a swagger-validator instance; used for the validation badge shown in swagger-ui. Can be modified to point to a local install of `swagger-validator` or set to None to remove the badge.

Default: `'http://online.swagger.io/validator/'` *Maps to parameter: validatorUrl*

OPERATIONS_SORTER

Sorting order for the operation list of each tag.

- None: show in the order returned by the server
- alpha: sort alphabetically by path
- method: sort by HTTP method

Default: None *Maps to parameter:* operationsSorter

TAGS_SORTER

Sorting order for tagged operation groups.

- None: Swagger UI default ordering
- alpha: sort alphabetically

Default: None *Maps to parameter:* tagsSorter

DOC_EXPANSION

Controls the default expansion setting for the operations and tags.

- None: everything is collapsed
- list: only tags are expanded
- full: all operations are expanded

Default: 'list' *Maps to parameter:* docExpansion

DEEP_LINKING

Automatically update the fragment part of the URL with permalinks to the currently selected operation.

Default: False *Maps to parameter:* deepLinking

SHOW_EXTENSIONS

Show vendor extension (x- . .) fields.

Default: True *Maps to parameter:* showExtensions

DEFAULT_MODEL_RENDERING

Controls whether operations show the model structure or the example value by default.

- model: show the model fields by default
- example: show the example value by default

Default: 'model' *Maps to parameter:* defaultModelRendering

DEFAULT_MODEL_DEPTH

Controls how many levels are expanded by default when showing nested models.

Default: 3 *Maps to parameter:* defaultModelExpandDepth

7.2 REDOC_SETTINGS

7.2.1 ReDoc UI settings

ReDoc UI configuration settings. See <https://github.com/Rebilly/ReDoc#redoc-tag-attributes>.

LAZY_RENDERING

Default: True *Maps to attribute:* lazy-rendering

HIDE_HOSTNAME

Default: False *Maps to attribute:* hide-hostname

EXPAND_RESPONSES

Default: 'all' *Maps to attribute:* expand-responses

PATH_IN_MIDDLE

Default: False *Maps to attribute:* path-in-middle-panel

Contributions are always welcome and appreciated! Here are some ways you can contribute.

8.1 Issues

You can and should open an issue for any of the following reasons:

- you found a bug; steps for reproducing, or a pull request with a failing test case will be greatly appreciated
- you wanted to do something but did not find a way to do it after reading the documentation
- you believe the current way of doing something is more complicated or less elegant than it can be
- a related feature that you want is missing from the package

Please always check for existing issues before opening a new issue.

8.2 Pull requests

You want to contribute some code? Great! Here are a few steps to get you started:

1. **Fork the repository on GitHub**
2. **Clone your fork and create a branch for the code you want to add**
3. **Create a new virtualenv and install the package in development mode**

```
$ virtualenv venv
$ source venv/bin/activate
(venv) $ pip install -e .[validation]
(venv) $ pip install -rrequirements/dev.txt "Django>=1.11.7"
```

4. **Make your changes and check them against the test project**

```
(venv) $ cd testproj
(venv) $ python manage.py migrate
(venv) $ python manage.py shell -c "import createsuperuser"
(venv) $ python manage.py runserver
(venv) $ firefox localhost:8000/swagger/
```

5. Update the tests if necessary

You can find them in the `tests` directory.

If your change modifies the expected schema output, you should regenerate the reference schema at `tests/reference.yaml`:

```
(venv) $ cd testproj
(venv) $ python manage.py generate_swagger ../tests/reference.yaml --overwrite --
↪user admin --url http://test.local:8002/
```

After checking the git diff to verify that no unexpected changes appeared, you should commit the new `reference.yaml` together with your changes.

6. Run tests. The project is setup to use tox and pytest for testing

```
# (optional) sort imports with isort and check flake8 linting
(venv) $ isort --apply
(venv) $ flake8 src/drf_yasg testproj tests setup.py
# run tests in the current environment, faster than tox
(venv) $ pytest --cov
# (optional) run tests for other python versions in separate environments
(venv) $ tox
```

7. Update documentation

If the change modifies behaviour or adds new features, you should update the documentation and `README.rst` accordingly. Documentation is written in reStructuredText and built using Sphinx. You can find the sources in the `docs` directory.

To build and check the docs, run

```
(venv) $ tox -e docs
```

8. Push your branch and submit a pull request to the master branch on GitHub

Incomplete/Work In Progress pull requests are encouraged, because they allow you to get feedback and help more easily.

9. Your code must pass all the required travis jobs before it is merged

As of now, this consists of running on Python 2.7, 3.4, 3.5 and 3.6, and building the docs successfully.

8.3 Maintainer's notes

8.3.1 Release checklist

- update `docs/changelog.rst` with changes since the last tagged version
- commit & tag the release - `git tag x.x.x -m "Release version x.x.x"`
- push using `git push --follow-tags`

- verify that [Travis](#) has built the tag and successfully published the release to [PyPI](#)
- publish release notes on [GitHub](#)
- start the [ReadTheDocs](#) build if it has not already started
- deploy the live demo on [Heroku](#)

9.1 BSD 3-Clause License

Copyright (c) 2017, Cristian V. <crisi@cvjd.me> All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

- Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
- Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
- Neither the name of the copyright holder nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS “AS IS” AND ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT HOLDER OR CONTRIBUTORS BE LIABLE FOR ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES (INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES; LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT (INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

10.1 1.3.1

- **FIXED:** fixed a bug that would sometimes cause endpoints to wrongly be output as form operations (#50)
- **IMPROVED:** added generation of `produces` based on renderer classes
- **IMPROVED:** added generation of top-level `consumes` and `produces` based on `DEFAULT_PARSER_CLASSES` and `DEFAULT_RENDERER_CLASSES` (#48)

10.2 1.3.0

- **ADDED:** security requirements are now correctly set and can be customized; this should fix problems related to authentication in `swagger-ui` Try it out! (#50, #54)
- **IMPROVED:** updated `swagger-ui` to version 3.9.2
- **IMPROVED:** updated `ReDoc` to version 1.20.0
- **FIXED:** fixed an exception caused by a warning in `get_path_from_regex` (#49, thanks to @blueyed)

10.3 1.2.2

- **FIXED:** `django>=3.7.7` is now required because of breaking changes (#44, #45, thanks to @hiro-kawa)

10.4 1.2.1

- Fixed deployment issues

10.5 1.2.0

- **ADDED:** `basePath` is now generated by taking into account the `SCRIPT_NAME` variable and the longest common prefix of API urls (#37, #42)
- **IMPROVED:** removed inline scripts and styles from bundled HTML templates to increase CSP compatibility
- **IMPROVED:** improved validation errors and added more assertion sanity checks (#37, #40)
- **IMPROVED:** improved handling of `NamespaceVersioning` by excluding endpoints of differing versions (i.e. when accessing the schema view for v1, v2 endpoints will not be included in swagger)

10.6 1.1.3

- **FIXED:** schema view cache will now always `Vary on the Cookie and Authentication` (the `Vary` header was previously only added if `public` was set to `True`) - this fixes issues related to Django authentication in `swagger-ui` and `CurrentUserDefault` values in the schema

10.7 1.1.2

- **IMPROVED:** updated `swagger-ui` to version 3.8.1
- **IMPROVED:** removed some unneeded static files

10.8 1.1.1

- **ADDED:** `generate_swagger_management_command` (#29, #31, thanks to @beaugunderson)
- **FIXED:** fixed improper generation of `\Z` regex tokens - will now be replaced by `$`

10.9 1.1.0

- **ADDED:** added support for APIs versioned with `URLPathVersioning` or `NamespaceVersioning`
- **ADDED:** added ability to recursively customize schema generation *using pluggable inspector classes*
- **ADDED:** added `operation_id` parameter to `@swagger_auto_schema`
- **ADDED:** integration with `django-rest-framework-camel-case` (#28)
- **IMPROVED:** strings, arrays and integers will now have min/max validation attributes inferred from the field-level validators
- **FIXED:** fixed a bug that caused `title` to never be generated for Schemas; `title` is now correctly populated from the field's `label` property

10.10 1.0.6

- **FIXED:** Swagger UI “Try it out!” should now work with Django login

- **FIXED:** callable default values on serializer fields will now be properly called (#24, #25)
- **IMPROVED:** updated `swagger-ui` to version 3.8.0
- **IMPROVED:** `PrimaryKeyRelatedField` and `SlugRelatedField` will now have appropriate types based on the related model (#26)
- **IMPROVED:** mock views will now have a bound request even with `public=False` (#23)

10.11 1.0.5

- **FIXED:** fixed a crash caused by having read-only Serializers nested by reference
- **FIXED:** removed erroneous backslashes in paths when routes are generated using Django 2 `path()`
- **IMPROVED:** updated `swagger-ui` to version 3.7.0
- **IMPROVED:** `FileField` is now generated as an URL or file name in response Schemas (#21, thanks to @h-hirokawa)

10.12 1.0.4

- **FIXED:** fixed improper generation of YAML references
- **ADDED:** added `query_serializer` parameter to `@swagger_auto_schema` (#16, #17)

10.13 1.0.3

- **FIXED:** fixed bug that caused schema views returned from cache to fail (#14)
- **FIXED:** disabled automatic generation of response schemas for form operations to avoid confusing errors caused by attempting to shove file parameters into Schema objects

10.14 1.0.2

- First published version

- [genindex](#)
- [modindex](#)
- [search](#)

11.1 drf_yasg package

11.1.1 drf_yasg.codecs

`drf_yasg.codecs._validate_flex` (*spec*)

`drf_yasg.codecs._validate_swagger_spec_validator` (*spec*)

`drf_yasg.codecs.VALIDATORS` = {'flex': <function _validate_flex>, 'ssv': <function _valida

class `drf_yasg.codecs._OpenAPICodec` (*validators*)

Bases: `object`

media_type = `None`

validators

List of validator names to apply

encode (*document*)

Transform an *Swagger* object to a sequence of bytes.

Also performs validation and applies settings.

Parameters *document* (`openapi.Swagger`) – Swagger spec object as generated by `OpenAPISchemaGenerator`

Returns binary encoding of document

Return type `bytes`

encode_error (*err*)

Dump an error message into an encoding-appropriate sequence of bytes

_dump_dict (*spec*)

Dump the given dictionary into its string representation.

Parameters *spec* (*dict*) – a python dict

Returns string representation of *spec*

Return type str

generate_swagger_object (*swagger*)

Generates the root Swagger object.

Parameters *swagger* (`openapi.Swagger`) – Swagger spec object as generated by `OpenAPISchemaGenerator`

Returns swagger spec as dict

Return type OrderedDict

class `drf_yasg.codecs.OpenAPICodecJson` (*validators*)

Bases: `drf_yasg.codecs._OpenAPICodec`

media_type = 'application/json'

_dump_dict (*spec*)

Dump spec into JSON.

`drf_yasg.codecs.yaml_sane_dump` (*data*, *binary*)

Dump the given data dictionary into a sane format:

- OrderedDicts are dumped as regular mappings instead of non-standard !!odict
- multi-line mapping style instead of json-like inline style
- list elements are indented into their parents
- YAML references/aliases are disabled

Parameters

- **data** (*dict*) – the data to be dumped

- **binary** (*bool*) – True to return a utf-8 encoded binary object, False to return a string

Returns the serialized YAML

Return type str,bytes

`drf_yasg.codecs.yaml_sane_load` (*stream*)

Load the given YAML stream while preserving the input order for mapping items.

Parameters *stream* – YAML stream (can be a string or a file-like object)

Return type OrderedDict

class `drf_yasg.codecs.OpenAPICodecYaml` (*validators*)

Bases: `drf_yasg.codecs._OpenAPICodec`

media_type = 'application/yaml'

_dump_dict (*spec*)

Dump spec into YAML.

11.1.2 drf_yasg.errors

exception `drf_yasg.errors.SwaggerError`

Bases: `Exception`

exception `drf_yasg.errors.SwaggerValidationError` (*msg*, *errors=None*, *spec=None*, *source_codec=None*, *args)

Bases: `drf_yasg.errors.SwaggerError`

exception `drf_yasg.errors.SwaggerGenerationError`

Bases: `drf_yasg.errors.SwaggerError`

11.1.3 drf_yasg.generators

class `drf_yasg.generators.EndpointEnumerator` (*patterns=None*, *urlconf=None*, *request=None*)

Bases: `rest_framework.schemas.generators.EndpointEnumerator`

get_path_from_regex (*path_regex*)

should_include_endpoint (*path*, *callback*, *app_name=""*, *namespace=""*, *url_name=None*)

replace_version (*path*, *callback*)

If `request.version` is not `None` and *callback* uses `URLPathVersioning`, this function replaces the `version` parameter in *path* with the actual version.

Parameters

- **path** (*str*) – the templated path
- **callback** – the view callback

Return type `str`

get_api_endpoints (*patterns=None*, *prefix=""*, *app_name=None*, *namespace=None*)

Return a list of all available API endpoints by inspecting the URL conf.

Copied entirely from super.

unescape (*s*)

Unescape all backslash escapes from *s*.

Parameters **s** (*str*) – string with backslash escapes

Return type `str`

unescape_path (*path*)

Remove backslash escapes from all path components outside {parameters}. This is needed because `simplify_regex` does not handle this correctly - note however that this implementation is

NOTE: this might destructively affect some url regex patterns that contain metacharacters (e.g. `w`, `d`) outside path parameter groups; if you are in this category, God help you

Parameters **path** (*str*) – path possibly containing

Returns the unescaped path

Return type `str`

class `drf_yasg.generators.OpenAPISchemaGenerator` (*info*, *version=""*, *url=None*, *patterns=None*, *urlconf=None*)

Bases: `object`

This class iterates over all registered API endpoints and returns an appropriate OpenAPI 2.0 compliant schema. Method implementations shamelessly stolen and adapted from rest-framework `SchemaGenerator`.

Parameters

- **info** (*Info*) – information about the API
- **version** (*str*) – API version string; if omitted, *info.default_version* will be used
- **url** (*str*) – API scheme, host and port; if `None` is passed and `DEFAULT_API_URL` is not set, the url will be inferred from the request made against the schema view, so you should generally not need to set this parameter explicitly; if the empty string is passed, no host and scheme will be emitted

If *url* is not `None` or the empty string, it must be a scheme-absolute uri (i.e. starting with `http://` or `https://`), and any path component is ignored;

See also: *documentation on base URL construction*

- **patterns** – if given, only these patterns will be enumerated for inclusion in the API spec
- **urlconf** – if patterns is not given, use this urlconf to enumerate patterns; if not given, the default urlconf is used

endpoint_enumerator_class

alias of *EndpointEnumerator*

url**get_schema** (*request=None, public=False*)

Generate a *Swagger* object representing the API schema.

Parameters

- **request** (*Request*) – the request used for filtering accessible endpoints and finding the spec URI
- **public** (*bool*) – if `True`, all endpoints are included regardless of access through *request*

Returns the generated Swagger specification

Return type *openapi.Swagger*

create_view (*callback, method, request=None*)

Create a view instance from a view callback as registered in `urlpatterns`.

Parameters

- **callback** (*callable*) – view callback registered in `urlpatterns`
- **method** (*str*) – HTTP method
- **request** (*rest_framework.request.Request*) – request to bind to the view

Returns the view instance

get_endpoints (*request*)

Iterate over all the registered endpoints in the API and return a fake view with the right parameters.

Parameters **request** (*rest_framework.request.Request*) – request to bind to the endpoint views

Returns {path: (view_class, list[(http_method, view_instance)])}

Return type dict

get_operation_keys (*subpath, method, view*)

Return a list of keys that should be used to group an operation within the specification.

```

/users/                ("users", "list"), ("users", "create")
/users/{pk}/          ("users", "read"), ("users", "update"), ("users",
↪"delete")
/users/enabled/       ("users", "enabled") # custom viewset list action
/users/{pk}/star/    ("users", "star") # custom viewset detail_
↪action
/users/{pk}/groups/  ("users", "groups", "list"), ("users", "groups",
↪"create")
/users/{pk}/groups/{pk}/ ("users", "groups", "read"), ("users", "groups",
↪"update")

```

Parameters

- **subpath** (*str*) – path to the operation with any common prefix/base path removed
- **method** (*str*) – HTTP method
- **view** – the view associated with the operation

Return type tuple**determine_path_prefix** (*paths*)

Given a list of all paths, return the common prefix which should be discounted when generating a schema structure.

This will be the longest common string that does not include that last component of the URL, or the last component before a path parameter.

For example:

```

/api/v1/users/
/api/v1/users/{pk}/

```

The path prefix is `/api/v1/`.

Parameters **paths** (*list[str]*) – list of paths

Return type str

get_paths (*endpoints, components, request, public*)

Generate the Swagger Paths for the API from the given endpoints.

Parameters

- **endpoints** (*dict*) – endpoints as returned by `get_endpoints`
- **components** (`ReferenceResolver`) – resolver/container for Swagger References
- **request** (`Request`) – the request made against the schema view; can be `None`
- **public** (*bool*) – if `True`, all endpoints are included regardless of access through *request*

Returns the `Paths` object and the longest common path prefix, as a 2-tuple

Return type tuple[`openapi.Paths`,str]

get_operation (*view, path, prefix, method, components, request*)

Get an *Operation* for the given API endpoint (path, method). This method delegates to *get_operation()* of a *ViewInspector* determined according to settings and *@swagger_auto_schema* overrides.

Parameters

- **view** – the view associated with this endpoint
- **path** (*str*) – the path component of the operation URL
- **prefix** (*str*) – common path prefix among all endpoints
- **method** (*str*) – the http method of the operation
- **components** (*openapi.ReferenceResolver*) – referenceable components
- **request** (*Request*) – the request made against the schema view; can be None

Return type *openapi.Operation*

get_path_item (*path, view_cls, operations*)

Get a *PathItem* object that describes the parameters and operations related to a single path in the API.

Parameters

- **path** (*str*) – the path
- **view_cls** (*type*) – the view that was bound to this path in urlpatterns
- **operations** (*dict[str, openapi.Operation]*) – operations defined on this path, keyed by lowercase HTTP method

Return type *openapi.PathItem*

get_overrides (*view, method*)

Get overrides specified for a given operation.

Parameters

- **view** – the view associated with the operation
- **method** (*str*) – HTTP method

Returns a dictionary containing any overrides set by *@swagger_auto_schema*

Return type dict

get_path_parameters (*path, view_cls*)

Return a list of *Parameter* instances corresponding to any templated path variables.

Parameters

- **path** (*str*) – templated request path
- **view_cls** (*type*) – the view class associated with the path

Returns path parameters

Return type list[*openapi.Parameter*]

11.1.4 drf_yasg.inspectors

`drf_yasg.inspectors.NotHandled = <object object>`

The most base type

class drf_yasg.inspectors.**BaseInspector** (*view, path, method, components, request*)

Bases: `object`

Parameters

- **view** – the view associated with this endpoint
- **path** (*str*) – the path component of the operation URL
- **method** (*str*) – the http method of the operation
- **components** (`openapi.ReferenceResolver`) – referenceable components
- **request** (*Request*) – the request made against the schema view; can be `None`

probe_inspectors (*inspectors, method_name, obj, initkwargs=None, **kwargs*)

Probe a list of inspectors with a given object. The first inspector in the list to return a value that is not `NotHandled` wins.

Parameters

- **inspectors** (*list [type [BaseInspector]]*) – list of inspectors to probe
- **method_name** (*str*) – name of the target method on the inspector
- **obj** – first argument to inspector method
- **initkwargs** (*dict*) – extra kwargs for instantiating inspector class
- **kwargs** – additional arguments to inspector method

Returns the return value of the winning inspector, or `None` if no inspector handled the object

process_result (*result, method_name, obj, **kwargs*)

After an inspector handles an object (i.e. returns a value other than `NotHandled`), all inspectors that were probed get the chance to alter the result, in reverse order. The inspector that handled the object is the first to receive a `process_result` call with the object it just returned.

This behaviour is similar to the Django request/response middleware processing.

If this inspector has no post-processing to do, it should just return `result` (the default implementation).

Parameters

- **result** – the return value of the winning inspector, or `None` if no inspector handled the object
- **method_name** (*str*) – name of the method that was called on the inspector
- **obj** – first argument passed to inspector method
- **kwargs** – additional arguments passed to inspector method

Returns

class drf_yasg.inspectors.**FilterInspector** (*view, path, method, components, request*)

Bases: `drf_yasg.inspectors.BaseInspector`

Base inspector for filter backends.

Responsible for determining extra query parameters added by given filter backends.

Parameters

- **view** – the view associated with this endpoint
- **path** (*str*) – the path component of the operation URL
- **method** (*str*) – the http method of the operation

- **components** (`openapi.ReferenceResolver`) – referenceable components
 - **request** (`Request`) – the request made against the schema view; can be `None`
- get_filter_parameters** (`filter_backend`)

Get the filter parameters for a single filter backend **instance**.

Should return *NotHandled* if this inspector does not know how to handle the given *filter_backend*.

Parameters `filter_backend` (`BaseFilterBackend`) – the filter backend

Return type `list[openapi.Parameter]`

class `drf_yasg.inspectors.PaginatorInspector` (`view`, `path`, `method`, `components`, `request`)

Bases: `drf_yasg.inspectors.BaseInspector`

Base inspector for paginators.

Responsible for determining extra query parameters and response structure added by given paginators.

Parameters

- **view** – the view associated with this endpoint
- **path** (`str`) – the path component of the operation URL
- **method** (`str`) – the http method of the operation
- **components** (`openapi.ReferenceResolver`) – referenceable components
- **request** (`Request`) – the request made against the schema view; can be `None`

get_paginated_response (`paginator`, `response_schema`)

Add appropriate paging fields to a response *Schema*.

Should return *NotHandled* if this inspector does not know how to handle the given *paginator*.

Parameters

- **paginator** (`BasePagination`) – the paginator
- **response_schema** (`openapi.Schema`) – the response schema that must be paged.

Return type `openapi.Schema`

get_paginator_parameters (`paginator`)

Get the pagination parameters for a single paginator **instance**.

Should return *NotHandled* if this inspector does not know how to handle the given *paginator*.

Parameters `paginator` (`BasePagination`) – the paginator

Return type `list[openapi.Parameter]`

class `drf_yasg.inspectors.FieldInspector` (`view`, `path`, `method`, `components`, `request`, `field_inspectors`)

Bases: `drf_yasg.inspectors.BaseInspector`

Base inspector for serializers and serializer fields.

_get_partial_types (`field`, `swagger_object_type`, `use_references`, `**kwargs`)

Helper method to extract generic information from a field and return a partial constructor for the appropriate openapi object.

All arguments are the same as `field_to_swagger_object()`.

The return value is a tuple consisting of:

- a function for constructing objects of `swagger_object_type`; its prototype is:

```
def SwaggerType(existing_object=None, **instance_kwargs):
```

This function creates an instance of *swagger_object_type*, passing the following attributes to its init, in order of precedence:

- arguments specified by the *kwargs* parameter of *_get_partial_types()*
- *instance_kwargs* passed to the constructor function
- *title*, *description*, *required* and *default* inferred from the field, where appropriate

If *existing_object* is not *None*, it is updated instead of creating a new object.

- a type that should be used for child objects if *field* is of an array type. This can currently have two values:
 - *Schema* if *swagger_object_type* is *Schema*
 - *Items* if *swagger_object_type* is *Parameter* or *Items*

Return type tuple[callable,(type[*openapi.Schema*],type[*openapi.Items*])]

field_to_swagger_object (*field*, *swagger_object_type*, *use_references*, ***kwargs*)

Convert a drf Serializer or Field instance into a Swagger object.

Should return *NotHandled* if this inspector does not know how to handle the given *field*.

Parameters

- **field** (*rest_framework.serializers.Field*) – the source field
- **swagger_object_type** (type[*openapi.SwaggerDict*]) – should be one of *Schema*, *Parameter*, *Items*
- **use_references** (*bool*) – if *False*, forces all objects to be declared inline instead of by referencing other components
- **kwargs** – extra attributes for constructing the object; if *swagger_object_type* is *Parameter*, *name* and *in_* should be provided

Returns the swagger object

Return type *openapi.Parameter*,*openapi.Items*,*openapi.Schema*,*openapi.SchemaRef*

probe_field_inspectors (*field*, *swagger_object_type*, *use_references*, ***kwargs*)

Helper method for recursively probing *field_inspectors* to handle a given field.

All arguments are the same as *field_to_swagger_object()*.

Return type *openapi.Parameter*,*openapi.Items*,*openapi.Schema*,*openapi.SchemaRef*

class drf_yasg.inspectors.**SerializerInspector** (*view*, *path*, *method*, *components*, *request*, *field_inspectors*)

Bases: *drf_yasg.inspectors.FieldInspector*

get_request_parameters (*serializer*, *in_*)

Convert a DRF serializer into a list of *Parameters*.

Should return *NotHandled* if this inspector does not know how to handle the given *serializer*.

Parameters

- **serializer** (*serializers.BaseSerializer*) – the *Serializer* instance

- `in` (*str*) – the location of the parameters, one of the *openapi.IN_** constants

Return type `list[openapi.Parameter]`

get_schema (*serializer*)

Convert a DRF Serializer instance to an *openapi.Schema*.

Should return *NotHandled* if this inspector does not know how to handle the given *serializer*.

Parameters `serializer` (*serializers.BaseSerializer*) – the Serializer instance

Return type *openapi.Schema*

class `drf_yasg.inspectors.ViewInspector` (*view, path, method, components, request, overrides*)

Bases: *drf_yasg.inspectors.BaseInspector*

Inspector class responsible for providing *Operation* definitions given a view, path and method.

Parameters `overrides` (*dict*) – manual overrides as passed to *@swagger_auto_schema_prepend_inspector_overrides* (*inspectors*)

`body_methods` = ('PUT', 'PATCH', 'POST')

`field_inspectors` = [`<class 'drf_yasg.inspectors.field.CamelCaseJSONFilter'>`, `<class 'd`

`filter_inspectors` = [`<class 'drf_yasg.inspectors.query.CoreAPICompatInspector'>`]

get_filter_parameters ()

Return the parameters added to the view by its filter backends.

Return type `list[openapi.Parameter]`

get_operation (*operation_keys*)

Get an *Operation* for the given API endpoint (path, method). This includes query, body parameters and response schemas.

Parameters `operation_keys` (*tuple[str]*) – an array of keys describing the hierarchical layout of this view in the API; e.g. ('snippets', 'list'), ('snippets', 'retrieve'), etc.

Return type *openapi.Operation*

get_paginated_response (*response_schema*)

Add appropriate paging fields to a response *Schema*.

Parameters `response_schema` (*openapi.Schema*) – the response schema that must be paged.

Returns the paginated response class: *Schema*, or *None* in case of an unknown pagination scheme

Return type *openapi.Schema*

get_pagination_parameters ()

Return the parameters added to the view by its paginator.

Return type `list[openapi.Parameter]`

`paginator_inspectors` = [`<class 'drf_yasg.inspectors.query.DjangoRestResponsePagination`

`serializer_to_parameters` (*serializer, in_*)

Convert a serializer to a possibly empty list of *Parameters*.

Parameters

- **serializer** (*serializers.BaseSerializer*) – the Serializer instance
- **in** (*str*) – the location of the parameters, one of the *openapi.IN_** constants

Return type list[*openapi.Parameter*]

serializer_to_schema (*serializer*)

Convert a serializer to an OpenAPI *Schema*.

Parameters **serializer** (*serializers.BaseSerializer*) – the Serializer instance

Returns the converted *Schema*, or None in case of an unknown serializer

Return type *openapi.Schema*,*openapi.SchemaRef*,None

should_filter ()

Determine whether filter backend parameters should be included for this request.

Return type bool

should_page ()

Determine whether paging parameters and structure should be added to this operation’s request and response.

Return type bool

class drf_yasg.inspectors.**CoreAPICompatInspector** (*view, path, method, components, request*)

Bases: *drf_yasg.inspectors.PaginatorInspector*, *drf_yasg.inspectors.FilterInspector*

Converts coreapi.Fields to *openapi.Parameters* for filters and paginators that implement a *get_schema_fields* method.

Parameters

- **view** – the view associated with this endpoint
- **path** (*str*) – the path component of the operation URL
- **method** (*str*) – the http method of the operation
- **components** (*openapi.ReferenceResolver*) – referenceable components
- **request** (*Request*) – the request made against the schema view; can be None

coreapi_field_to_parameter (*field*)

Convert an instance of *coreapi.Field* to a swagger *Parameter* object.

Parameters **field** (*coreapi.Field*) –

Return type *openapi.Parameter*

get_filter_parameters (*filter_backend*)

get_paginator_parameters (*paginator*)

class drf_yasg.inspectors.**DjangoRestResponsePagination** (*view, path, method, components, request*)

Bases: *drf_yasg.inspectors.PaginatorInspector*

Provides response schema pagination warpping for django-rest-framework’s *LimitOffsetPagination*, *PageNumberPagination* and *CursorPagination*

Parameters

- **view** – the view associated with this endpoint

- **path** (*str*) – the path component of the operation URL
- **method** (*str*) – the http method of the operation
- **components** (*openapi.ReferenceResolver*) – referenceable components
- **request** (*Request*) – the request made against the schema view; can be None

get_paginated_response (*paginator, response_schema*)

class `drf_yasg.inspectors.InlineSerializerInspector` (*view, path, method, components, request, field_inspectors*)

Bases: `drf_yasg.inspectors.SerializerInspector`

Provides serializer conversions using `FieldInspector.field_to_swagger_object()`.

field_to_swagger_object (*field, swagger_object_type, use_references, **kwargs*)

get_parameter_name (*field_name*)

get_property_name (*field_name*)

get_request_parameters (*serializer, in_*)

get_schema (*serializer*)

use_definitions = False

class `drf_yasg.inspectors.ReferencingSerializerInspector` (*view, path, method, components, request, field_inspectors*)

Bases: `drf_yasg.inspectors.InlineSerializerInspector`

use_definitions = True

class `drf_yasg.inspectors.RelatedFieldInspector` (*view, path, method, components, request, field_inspectors*)

Bases: `drf_yasg.inspectors.FieldInspector`

Provides conversions for RelatedFields.

field_to_swagger_object (*field, swagger_object_type, use_references, **kwargs*)

class `drf_yasg.inspectors.SimpleFieldInspector` (*view, path, method, components, request, field_inspectors*)

Bases: `drf_yasg.inspectors.FieldInspector`

Provides conversions for fields which can be described using just type, format, pattern and min/max validators.

field_to_swagger_object (*field, swagger_object_type, use_references, **kwargs*)

class `drf_yasg.inspectors.FileFieldInspector` (*view, path, method, components, request, field_inspectors*)

Bases: `drf_yasg.inspectors.FieldInspector`

Provides conversions for FileFields.

field_to_swagger_object (*field, swagger_object_type, use_references, **kwargs*)

class `drf_yasg.inspectors.ChoiceFieldInspector` (*view, path, method, components, request, field_inspectors*)

Bases: `drf_yasg.inspectors.FieldInspector`

Provides conversions for ChoiceField and MultipleChoiceField.

field_to_swagger_object (*field, swagger_object_type, use_references, **kwargs*)

```

class drf_yasg.inspectors.DictFieldInspector (view, path, method, components, request,
                                             field_inspectors)
    Bases: drf_yasg.inspectors.FieldInspector
    Provides conversion for DictField.
    field_to_swagger_object (field, swagger_object_type, use_references, **kwargs)

class drf_yasg.inspectors.StringDefaultFieldInspector (view, path, method,
                                                       components, request,
                                                       field_inspectors)
    Bases: drf_yasg.inspectors.FieldInspector
    For otherwise unhandled fields, return them as plain TYPE_STRING objects.
    field_to_swagger_object (field, swagger_object_type, use_references, **kwargs)

class drf_yasg.inspectors.CamelCaseJSONFilter (view, path, method, components, request,
                                               field_inspectors)
    Bases: drf_yasg.inspectors.FieldInspector
    Converts property names to camelCase if CamelCaseJSONParser or CamelCaseJSONRenderer are
    used.
    is_camel_case ()
    process_result (result, method_name, obj, **kwargs)

class drf_yasg.inspectors.SwaggerAutoSchema (view, path, method, components, request,
                                             overrides)
    Bases: drf_yasg.inspectors.ViewInspector
    add_manual_parameters (parameters)
        Add/replace parameters from the given list of automatically generated request parameters.
        Parameters parameters (list [openapi.Parameter]) – generated parameters
        Returns modified parameters
        Return type list[openapi.Parameter]

    get_consumes ()
        Return the MIME types this endpoint can consume.
        Return type list[str]

    get_default_responses ()
        Get the default responses determined for this view from the request serializer and request method.
        Type dict[str, openapi.Schema]

    get_description ()
        Return an operation description determined as appropriate from the view’s method and class docstrings.
        Returns the operation description
        Return type str

    get_operation (operation_keys)

    get_operation_id (operation_keys)
        Return an unique ID for this operation. The ID must be unique across all Operation objects in the API.
        Parameters operation_keys (tuple[str]) – an array of keys derived from the
        pathdescribing the hierarchical layout of this view in the API; e.g. ('snippets',
        'list'), ('snippets', 'retrieve'), etc.

```

Return type str

get_produces ()

Return the MIME types this endpoint can produce.

Return type list[str]

get_query_parameters ()

Return the query parameters accepted by this view.

Return type list[*openapi.Parameter*]

get_query_serializer ()

Return the query serializer (used for parsing query parameters) for this endpoint.

Returns the query serializer, or None

get_request_body_parameters (*consumes*)

Return the request body parameters for this view. This is either:

- a list with a single object *Parameter* with a *Schema* derived from the request serializer
- a list of primitive *Parameters* parsed as form data

Parameters consumes (*list[str]*) – a list of accepted MIME types as returned by *get_consumes* ()

Returns a (potentially empty) list of *Parameters* either *in: body* or *in: formData*

Return type list[*openapi.Parameter*]

get_request_body_schema (*serializer*)

Return the *Schema* for a given request's body data. Only applies to PUT, PATCH and POST requests.

Parameters serializer – the view's request serializer as returned by *get_request_serializer* ()

Return type *openapi.Schema*

get_request_form_parameters (*serializer*)

Given a *Serializer*, return a list of *in: formData Parameters*.

Parameters serializer – the view's request serializer as returned by *get_request_serializer* ()

Return type list[*openapi.Parameter*]

get_request_serializer ()

Return the request serializer (used for parsing the request payload) for this endpoint.

Returns the request serializer, or one of *Schema*, *SchemaRef*, None

get_response_schemas (*response_serializers*)

Return the *openapi.Response* objects calculated for this view.

Parameters response_serializers (*dict*) – response serializers as returned by *get_response_serializers* ()

Returns a dictionary of status code to *Response* object

Return type dict[str, *openapi.Response*]

get_response_serializers()

Return the response codes that this view is expected to return, and the serializer for each response body. The return value should be a dict where the keys are possible status codes, and values are either strings, Serializers, *Schema*, *SchemaRef* or *Response* objects. See *@swagger_auto_schema* for more details.

Returns the response serializers

Return type dict

get_responses()

Get the possible responses for this view as a swagger *Responses* object.

Returns the documented responses

Return type *openapi.Responses*

get_security()

Return a list of security requirements for this operation.

Returning an empty list marks the endpoint as unauthenticated (i.e. removes all accepted authentication schemes). Returning *None* will inherit the top-level security requirements.

Returns security requirements

Return type list[dict[str,list[str]]]

get_tags(*operation_keys*)

Get a list of tags for this operation. Tags determine how operations relate with each other, and in the UI each tag will show as a group containing the operations that use it.

Parameters **operation_keys** (*tuple[str]*) – an array of keys derived from the path describing the hierarchical layout of this view in the API; e.g. ('snippets', 'list'), ('snippets', 'retrieve'), etc.

Return type list[str]

get_view_serializer()

Return the serializer as defined by the view's *get_serializer()* method.

Returns the view's *Serializer*

make_body_parameter(*schema*)

Given a *Schema* object, create an in: *body Parameter*.

Parameters **schema** (*openapi.Schema*) – the request body schema

Return type *openapi.Parameter*

11.1.5 drf_yasg.middleware

class drf_yasg.middleware.SwaggerExceptionMiddleware (*get_response*)

Bases: object

process_exception (*request, exception*)

11.1.6 drf_yasg.openapi

drf_yasg.openapi.TYPE_OBJECT = 'object'

drf_yasg.openapi.TYPE_STRING = 'string'

```
drf_yasg.openapi.TYPE_NUMBER = 'number'
drf_yasg.openapi.TYPE_INTEGER = 'integer'
drf_yasg.openapi.TYPE_BOOLEAN = 'boolean'
drf_yasg.openapi.TYPE_ARRAY = 'array'
drf_yasg.openapi.TYPE_FILE = 'file'
drf_yasg.openapi.FORMAT_DATE = 'date'
drf_yasg.openapi.FORMAT_DATETIME = 'date-time'
drf_yasg.openapi.FORMAT_PASSWORD = 'password'
drf_yasg.openapi.FORMAT_BINARY = 'binary'
drf_yasg.openapi.FORMAT_BASE64 = 'bytes'
drf_yasg.openapi.FORMAT_FLOAT = 'float'
drf_yasg.openapi.FORMAT_DOUBLE = 'double'
drf_yasg.openapi.FORMAT_INT32 = 'int32'
drf_yasg.openapi.FORMAT_INT64 = 'int64'
drf_yasg.openapi.FORMAT_EMAIL = 'email'
drf_yasg.openapi.FORMAT_IPV4 = 'ipv4'
drf_yasg.openapi.FORMAT_IPV6 = 'ipv6'
drf_yasg.openapi.FORMAT_URI = 'uri'
drf_yasg.openapi.FORMAT_UUID = 'uuid'
drf_yasg.openapi.FORMAT_SLUG = 'slug'
drf_yasg.openapi.IN_BODY = 'body'
drf_yasg.openapi.IN_PATH = 'path'
drf_yasg.openapi.IN_QUERY = 'query'
drf_yasg.openapi.IN_FORM = 'formData'
drf_yasg.openapi.IN_HEADER = 'header'
drf_yasg.openapi.SCHEMA_DEFINITIONS = 'definitions'
```

```
drf_yasg.openapi.make_swagger_name(attribute_name)
```

Convert a python variable name into a Swagger spec attribute name.

In particular,

- if name starts with `x_`, return `x-{camelCase}`
- if name is `ref`, return `$ref`
- else return the name converted to camelCase, with trailing underscores stripped

Parameters `attribute_name` (*str*) – python attribute name

Returns swagger name

```
class drf_yasg.openapi.SwaggerDict(**attrs)
```

Bases: `collections.OrderedDict`

A particular type of `OrderedDict`, which maps all attribute accesses to dict lookups using `make_swagger_name()`. Attribute names starting with `_` are set on the object as-is and are not included in the specification output.

Used as a base class for all Swagger helper models.

`__insert_extras__()`

From an ordering perspective, it is desired that extra attributes such as vendor extensions stay at the bottom of the object. However, python2.7's `OrderedDict` craps out if you try to insert into it before calling `init`. This means that subclasses must call `super().__init__` as the first statement of their own `__init__`, which would result in the extra attributes being added first. For this reason, we defer the insertion of the attributes and require that subclasses call `__insert_extras__` at the end of their `__init__` method.

`static _as_odict(obj, memo)`

Implementation detail of `as_odict()`

`as_odict()`

Convert this object into an `OrderedDict` instance.

Return type `OrderedDict`

`class drf_yasg.openapi.Contact` (*name=None, url=None, email=None, **extra*)

Bases: `drf_yasg.openapi.SwaggerDict`

Swagger Contact object

At least one of the following fields is required:

Parameters

- **`name`** (*str*) – contact name
- **`url`** (*str*) – contact url
- **`email`** (*str*) – contact e-mail

`class drf_yasg.openapi.License` (*name, url=None, **extra*)

Bases: `drf_yasg.openapi.SwaggerDict`

Swagger License object

Parameters

- **`name`** (*str*) – Required. License name
- **`url`** (*str*) – link to detailed license information

`class drf_yasg.openapi.Info` (*title, default_version, description=None, terms_of_service=None, contact=None, license=None, **extra*)

Bases: `drf_yasg.openapi.SwaggerDict`

Swagger Info object

Parameters

- **`title`** (*str*) – Required. API title.
- **`default_version`** (*str*) – Required. API version string (not to be confused with Swagger spec version)
- **`description`** (*str*) – API description; markdown supported
- **`terms_of_service`** (*str*) – API terms of service; should be a URL
- **`contact`** (`Contact`) – contact object
- **`license`** (`License`) – license object

```
class drf_yasg.openapi.Swagger (info=None, _url=None, _prefix=None, _version=None, consumes=None, produces=None, security_definitions=None, security=None, paths=None, definitions=None, **extra)
```

Bases: *drf_yasg.openapi.SwaggerDict*

Root Swagger object.

Parameters

- **info** (*Info*) – info object
- **_url** (*str*) – URL used for setting the API host and scheme
- **_prefix** (*str*) – api path prefix to use in setting basePath; this will be appended to the wsgi SCRIPT_NAME prefix or Django’s FORCE_SCRIPT_NAME if applicable
- **_version** (*str*) – version string to override Info
- **security_definitions** (*list[dict]*) – list of supported authentication mechanisms
- **security** (*list[dict]*) – authentication mechanisms accepted by default; can be overridden in Operation
- **consumes** (*list[str]*) – consumed MIME types; can be overridden in Operation
- **produces** (*list[str]*) – produced MIME types; can be overridden in Operation
- **paths** (*Paths*) – paths object
- **definitions** (*dict[str, Schema]*) – named models

```
classmethod get_base_path (script_prefix, api_prefix)
```

Determine an appropriate value for basePath based on the SCRIPT_NAME and the api common prefix.

Parameters

- **script_prefix** (*str*) – script prefix as defined by django get_script_prefix
- **api_prefix** (*str*) – api common prefix

Returns joined base path

```
class drf_yasg.openapi.Paths (paths, **extra)
```

Bases: *drf_yasg.openapi.SwaggerDict*

A listing of all the paths in the API.

Parameters **paths** (*dict[str, PathItem]*) –

```
class drf_yasg.openapi.PathItem (get=None, put=None, post=None, delete=None, options=None, head=None, patch=None, parameters=None, **extra)
```

Bases: *drf_yasg.openapi.SwaggerDict*

Information about a single path

Parameters

- **get** (*Operation*) – operation for GET
- **put** (*Operation*) – operation for PUT
- **post** (*Operation*) – operation for POST
- **delete** (*Operation*) – operation for DELETE
- **options** (*Operation*) – operation for OPTIONS
- **head** (*Operation*) – operation for HEAD

- **patch** (*Operation*) – operation for PATCH
- **parameters** (*list [Parameter]*) – parameters that apply to all operations

```
class drf_yasg.openapi.Operation (operation_id, responses, parameters=None, consumes=None,
                                produces=None, summary=None, description=None,
                                tags=None, security=None, **extra)
```

Bases: *drf_yasg.openapi.SwaggerDict*

Information about an API operation (path + http method combination)

Parameters

- **operation_id** (*str*) – operation ID, should be unique across all operations
- **responses** (*Responses*) – responses returned
- **parameters** (*list [Parameter]*) – parameters accepted
- **consumes** (*list [str]*) – content types accepted
- **produces** (*list [str]*) – content types produced
- **summary** (*str*) – operation summary; should be < 120 characters
- **description** (*str*) – operation description; can be of any length and supports mark-down
- **tags** (*list [str]*) – operation tags
- **security** (*list [dict [str, list [str]]]*) – list of security requirements

```
class drf_yasg.openapi.Items (type=None, format=None, enum=None, pattern=None,
                              items=None, **extra)
```

Bases: *drf_yasg.openapi.SwaggerDict*

Used when defining an array *Parameter* to describe the array elements.

Parameters

- **type** (*str*) – type of the array elements; must not be `object`
- **format** (*str*) – value format, see OpenAPI spec
- **enum** (*list*) – restrict possible values
- **pattern** (*str*) – pattern if type is `string`
- **items** (*Items*) – only valid if *type* is `array`

```
class drf_yasg.openapi.Parameter (name, in_, description=None, required=None,
                                   schema=None, type=None, format=None, enum=None,
                                   pattern=None, items=None, default=None, **extra)
```

Bases: *drf_yasg.openapi.SwaggerDict*

Describe parameters accepted by an *Operation*. Each parameter should be a unique combination of (*name*, *in_*). `body` and `form` parameters in the same operation are mutually exclusive.

Parameters

- **name** (*str*) – parameter name
- **in** (*str*) – parameter location
- **description** (*str*) – parameter description
- **required** (*bool*) – whether the parameter is required for the operation
- **schema** (*Schema, SchemaRef*) – required if *in_* is `body`
- **type** (*str*) – parameter type; required if *in_* is not `body`; must not be `object`

- **format** (*str*) – value format, see OpenAPI spec
- **enum** (*list*) – restrict possible values
- **pattern** (*str*) – pattern if type is `string`
- **items** (`Items`) – only valid if *type* is `array`
- **default** – default value if the parameter is not provided; must conform to parameter type

```
class drf_yasg.openapi.Schema (title=None, description=None, type=None, format=None,
                               enum=None, pattern=None, properties=None, additional_properties=None,
                               required=None, items=None, default=None, read_only=None, **extra)
```

Bases: `drf_yasg.openapi.SwaggerDict`

Describes a complex object accepted as parameter or returned as a response.

Parameters

- **title** (*str*) – schema title
- **description** (*str*) – schema description
- **type** (*str*) – value type; required
- **format** (*str*) – value format, see OpenAPI spec
- **enum** (*list*) – restrict possible values
- **pattern** (*str*) – pattern if type is `string`
- **properties** (*list* [`Schema`, `SchemaRef`]) – object properties; required if *type* is `object`
- **additional_properties** (*bool*, `Schema`, `SchemaRef`) – allow wildcard properties not listed in *properties*
- **required** (*list* [*str*]) – list of required property names
- **items** (`Schema`, `SchemaRef`) – type of array items, only valid if *type* is `array`
- **default** – only valid when insider another `Schema`'s *properties*; the default value of this property if it is not provided, must conform to the type of this `Schema`
- **read_only** – only valid when insider another `Schema`'s *properties*; declares the property as read only - it must only be sent as part of responses, never in requests

```
OR_REF = (<class 'drf_yasg.openapi.Schema'>, <class 'drf_yasg.openapi.SchemaRef'>)  
useful for type-checking, e.g isinstance(obj,.openapi.Schema.OR_REF)
```

```
class drf_yasg.openapi._Ref (resolver, name, scope, expected_type)
```

Bases: `drf_yasg.openapi.SwaggerDict`

Base class for all reference types. A reference object has only one property, `$ref`, which must be a JSON reference to a valid object in the specification, e.g. `#/definitions/Article` to refer to an article model.

Parameters

- **resolver** (`ReferenceResolver`) – component resolver which must contain the referenced object
- **name** (*str*) – referenced object name, e.g. “Article”
- **scope** (*str*) – reference scope, e.g. “definitions”
- **expected_type** (*type* [`SwaggerDict`]) – the expected type that will be asserted on the object found in resolver


```
ref_name_re = re.compile('#/(?P<scope>.+)/(?P<name>[^\/]*)$')
```

resolve (*resolver*)

Get the object targeted by this reference from the given component resolver.

Parameters **resolver** (*ReferenceResolver*) – component resolver which must contain the referenced object

Returns the target object

class `drf_yasg.openapi.SchemaRef` (*resolver, schema_name*)

Bases: `drf_yasg.openapi._Ref`

Adds a reference to a named Schema defined in the `#/definitions/` object.

Parameters

- **resolver** (*ReferenceResolver*) – component resolver which must contain the definition
- **schema_name** (*str*) – schema name

`drf_yasg.openapi.resolve_ref` (*ref_or_obj, resolver*)

Resolve *ref_or_obj* if it is a reference type. Return it unchanged if not.

Parameters

- **ref_or_obj** (*SwaggerDict, _Ref*) –
- **resolver** – component resolver which must contain the referenced object

class `drf_yasg.openapi.Responses` (*responses, default=None, **extra*)

Bases: `drf_yasg.openapi.SwaggerDict`

Describes the expected responses of an *Operation*.

Parameters

- **responses** (*dict[(str, int), Response]*) – mapping of status code to response definition
- **default** (*Response*) – description of the response structure to expect if another status code is returned

class `drf_yasg.openapi.Response` (*description, schema=None, examples=None, **extra*)

Bases: `drf_yasg.openapi.SwaggerDict`

Describes the structure of an operation's response.

Parameters

- **description** (*str*) – response description
- **schema** (*Schema, SchemaRef*) – structure of the response body
- **examples** (*dict*) – example bodies mapped by mime type

class `drf_yasg.openapi.ReferenceResolver` (**scopes*)

Bases: `object`

A mapping type intended for storing objects pointed at by Swagger Refs. Provides support and checks for different reference scopes, e.g. 'definitions'.

For example:

```
> components = ReferenceResolver('definitions', 'parameters')
> definitions = ReferenceResolver.with_scope('definitions')
> definitions.set('Article', Schema(...))
```

```
> print(components)
{'definitions': OrderedDict([('Article', Schema(...)]), 'parameters':
↳OrderedDict() }
```

Parameters `scopes` (*str*) – an enumeration of the valid scopes this resolver will contain

with_scope (*scope*)

Return a view into this *ReferenceResolver* whose scope is defaulted and forced to *scope*.

Parameters `scope` (*str*) – target scope, must be in this resolver's *scopes*

Returns the bound resolver

Return type *ReferenceResolver*

_check_scope (*scope*)

set (*name*, *obj*, *scope=None*)

Set an object in the given scope, raise an error if it already exists.

Parameters

- **name** (*str*) – reference name
- **obj** – referenced object
- **scope** (*str*) – reference scope

setdefault (*name*, *maker*, *scope=None*)

Set an object in the given scope only if it does not exist.

Parameters

- **name** (*str*) – reference name
- **maker** (*callable*) – object factory, called only if necessary
- **scope** (*str*) – reference scope

get (*name*, *scope=None*)

Get an object from the given scope, raise an error if it does not exist.

Parameters

- **name** (*str*) – reference name
- **scope** (*str*) – reference scope

Returns the object

getdefault (*name*, *default=None*, *scope=None*)

Get an object from the given scope or a default value if it does not exist.

Parameters

- **name** (*str*) – reference name
- **default** – the default value
- **scope** (*str*) – reference scope

Returns the object or *default*

has (*name*, *scope=None*)

Check if an object exists in the given scope.

Parameters

- **name** (*str*) – reference name
- **scope** (*str*) – reference scope

Returns True if the object exists

Return type bool

scopes

keys ()

11.1.7 drf_yasg.renderers

class drf_yasg.renderers._SpecRenderer

Bases: rest_framework.renderers.BaseRenderer

Base class for text renderers. Handles encoding and validation.

charset = None

validators = []

codec_class = None

classmethod with_validators (*validators*)

render (*data*, *media_type=None*, *renderer_context=None*)

class drf_yasg.renderers.OpenAPIRenderer

Bases: *drf_yasg.renderers._SpecRenderer*

Renders the schema as a JSON document with the application/openapi+json specific mime type.

media_type = 'application/openapi+json'

format = 'openapi'

codec_class

alias of OpenAPICodecJson

class drf_yasg.renderers.SwaggerJSONRenderer

Bases: *drf_yasg.renderers._SpecRenderer*

Renders the schema as a JSON document with the generic application/json mime type.

media_type = 'application/json'

format = '.json'

codec_class

alias of OpenAPICodecJson

class drf_yasg.renderers.SwaggerYAMLRenderer

Bases: *drf_yasg.renderers._SpecRenderer*

Renders the schema as a YAML document.

media_type = 'application/yaml'

format = '.yaml'

codec_class

alias of OpenAPICodecYaml

class drf_yasg.renderers._UIRenderer

Bases: `rest_framework.renderers.BaseRenderer`

Base class for web UI renderers. Handles loading and passing settings to the appropriate template.

`media_type = 'text/html'`

`charset = 'utf-8'`

`template = ''`

`render (swagger, accepted_media_type=None, renderer_context=None)`

`set_context (renderer_context, swagger)`

`get_auth_urls ()`

`get_swagger_ui_settings ()`

`get_redoc_settings ()`

class drf_yasg.renderers.SwaggerUIRenderer

Bases: `drf_yasg.renderers._UIRenderer`

Renders a swagger-ui web interface for schema browsing. Also requires `OpenAPIRenderer` as an available renderer on the same view.

`template = 'drf-yasg/swagger-ui.html'`

`format = 'swagger'`

class drf_yasg.renderers.ReDocRenderer

Bases: `drf_yasg.renderers._UIRenderer`

Renders a ReDoc web interface for schema browsing. Also requires `OpenAPIRenderer` as an available renderer on the same view.

`template = 'drf-yasg/redoc.html'`

`format = 'redoc'`

class drf_yasg.renderers.ReDocAlphaRenderer

Bases: `drf_yasg.renderers._UIRenderer`

Renders a ReDoc web interface for schema browsing. Also requires `OpenAPIRenderer` as an available renderer on the same view.

`template = 'drf-yasg/redoc-alpha.html'`

`format = 'redoc'`

11.1.8 drf_yasg.utils

`drf_yasg.utils.no_body = <object object>`

used to forcibly remove the body of a request via `swagger_auto_schema ()`

`drf_yasg.utils.swagger_auto_schema (method=None, methods=None, auto_schema=<object object>, request_body=None, query_serializer=None, manual_parameters=None, operation_id=None, operation_description=None, security=None, responses=None, field_inspectors=None, filter_inspectors=None, paginator_inspectors=None, **extra_overrides)`

Decorate a view method to customize the `Operation` object generated from it.

method and *methods* are mutually exclusive and must only be present when decorating a view method that accepts more than one HTTP request method.

The *auto_schema* and *operation_description* arguments take precedence over view- or method-level values.

Parameters

- **method** (*str*) – for multi-method views, the http method the options should apply to
- **methods** (*list[str]*) – for multi-method views, the http methods the options should apply to
- **auto_schema** (*inspectors.SwaggerAutoSchema*) – custom class to use for generating the Operation object; this overrides both the class-level *swagger_schema* attribute and the *DEFAULT_AUTO_SCHEMA_CLASS* setting, and can be set to *None* to prevent this operation from being generated

- **request_body** (*Schema, SchemaRef, Serializer*) – custom request body, or *no_body*. The value given here will be used as the *schema* property of a *Parameter* with *in: 'body'*.

A *Schema* or *SchemaRef* is not valid if this request consumes form-data, because *form* and *body* parameters are mutually exclusive in an *Operation*. If you need to set custom *form* parameters, you can use the *manual_parameters* argument.

If a *Serializer* class or instance is given, it will be automatically converted into a *Schema* used as a *body Parameter*, or into a list of *form Parameters*, as appropriate.

- **query_serializer** (*Serializer*) – if you use a *Serializer* to parse query parameters, you can pass it here and have *Parameter* objects be generated automatically from it.

If any *Field* on the serializer cannot be represented as a *query Parameter* (e.g. nested Serializers, file fields, ...), the schema generation will fail with an error.

Schema generation will also fail if the name of any *Field* on the *query_serializer* conflicts with parameters generated by *filter_backends* or *paginator*.

- **manual_parameters** (*list[Parameter]*) – a list of manual parameters to override the automatically generated ones

Parameters are identified by their (name, in) combination, and any parameters given here will fully override automatically generated parameters if they collide.

It is an error to supply *form* parameters when the request does not consume form-data.

- **operation_id** (*str*) – operation ID override; the operation ID must be unique across the whole API
- **operation_description** (*str*) – operation description override
- **security** (*str*) – security requirements override; used to specify which authentication mechanism is required to call this API; an empty list marks the endpoint as unauthenticated (i.e. removes all accepted authentication schemes), and *None* will inherit the top-level security requirements
- **responses** (*dict[str, (Schema, SchemaRef, Response, str, Serializer)]*) – a dict of documented manual responses keyed on response status code. If no success (2xx) response is given, one will automatically be generated from the request body and http method. If any 2xx response is given the automatic response is suppressed.

- if a plain string is given as value, a *Response* with no body and that string as its description will be generated
- if a *Schema*, *SchemaRef* is given, a *Response* with the schema as its body and an empty description will be generated
- a *Serializer* class or instance will be converted into a *Schema* and treated as above
- a *Response* object will be used as-is; however if its *schema* attribute is a *Serializer*, it will automatically be converted into a *Schema*
- **field_inspectors** (*list* [*FieldInspector*]) – extra serializer and field inspectors; these will be tried before *ViewInspector.field_inspectors* on the *inspectors.SwaggerAutoSchema* instance
- **filter_inspectors** (*list* [*FilterInspector*]) – extra filter inspectors; these will be tried before *ViewInspector.filter_inspectors* on the *inspectors.SwaggerAutoSchema* instance
- **paginator_inspectors** (*list* [*PaginatorInspector*]) – extra paginator inspectors; these will be tried before *ViewInspector.paginator_inspectors* on the *inspectors.SwaggerAutoSchema* instance
- **extra_overrides** – extra values that will be saved into the overrides dict; these values will be available in the handling *inspectors.SwaggerAutoSchema* instance via *self.overrides*

`drf_yasg.utils.is_list_view(path, method, view)`

Check if the given path/method appears to represent a list view (as opposed to a detail/instance view).

Parameters

- **path** (*str*) – view path
- **method** (*str*) – http method
- **view** (*APIView*) – target view

Return type bool

`drf_yasg.utils.guess_response_status(method)`

`drf_yasg.utils.param_list_to_odict(parameters)`

Transform a list of *Parameter* objects into an *OrderedDict* keyed on the (name, in_) tuple of each parameter.

Raises an *AssertionError* if *parameters* contains duplicate parameters (by their name + in combination).

Parameters *parameters* (*list* [*Parameter*]) – the list of parameters

Returns *parameters* keyed by (name, in_)

Return type dict[tuple(str,str),*Parameter*]

`drf_yasg.utils.filter_none(obj)`

Remove *None* values from tuples, lists or dictionaries. Return other objects as-is.

Parameters *obj* –

Returns collection with *None* values removed

`drf_yasg.utils.force_serializer_instance(serializer)`

Force *serializer* into a *Serializer* instance. If it is not a *Serializer* class or instance, raises an *AssertionError*.

Parameters *serializer* – serializer class or instance

Returns serializer instance

`drf_yasg.utils.get_consumes(parser_classes)`

Extract consumes MIME types from a list of parser classes.

Parameters `parser_classes` (*list*) – parser classes

Returns MIME types for consumes

Return type list[str]

`drf_yasg.utils.get_produces(renderer_classes)`

Extract produces MIME types from a list of renderer classes.

Parameters `renderer_classes` (*list*) – renderer classes

Returns MIME types for produces

Return type list[str]

11.1.9 drf_yasg.views

`drf_yasg.views.deferred_never_cache(view_func)`

Decorator that adds headers to a response so that it will never be cached.

`drf_yasg.views.get_schema_view(info=None, url=None, patterns=None, urlconf=None, public=False, validators=None, generator_class=<class 'drf_yasg.generators.OpenAPISchemaGenerator'>, authentication_classes=[<class 'rest_framework.authentication.SessionAuthentication'>, <class 'rest_framework.authentication.BasicAuthentication'>], permission_classes=[<class 'rest_framework.permissions.AllowAny'>])`

Create a SchemaView class with default renderers and generators.

Parameters

- **info** (*Info*) – information about the API; if omitted, defaults to `DEFAULT_INFO`
- **url** (*str*) – same as `OpenAPISchemaGenerator`
- **patterns** – same as `OpenAPISchemaGenerator`
- **urlconf** – same as `OpenAPISchemaGenerator`
- **public** (*bool*) – if False, includes only the endpoints that are accessible by the user viewing the schema
- **validators** (*list*) – a list of validator names to apply; allowed values are `flex`, `ssv`
- **generator_class** (*type*) – schema generator class to use; should be a subclass of `OpenAPISchemaGenerator`
- **authentication_classes** (*tuple*) – authentication classes for the schema view itself
- **permission_classes** (*tuple*) – permission classes for the schema view itself

Returns SchemaView class

Return type type[`SchemaView`]

`class drf_yasg.views.SchemaView(**kwargs)`

Bases: `rest_framework.views.APIView`

Constructor. Called in the URLconf; can contain helpful extra keyword arguments, and other things.

`_ignore_model_permissions = True`

classmethod `apply_cache` (*view*, *cache_timeout*, *cache_kwargs*)

Override this method to customize how caching is applied to the view.

Arguments described in `as_cached_view()`.

classmethod `as_cached_view` (*cache_timeout=0*, *cache_kwargs=None*, ***initkwargs*)

Calls `.as_view()` and wraps the result in a `cache_page` decorator. See <https://docs.djangoproject.com/en/1.11/topics/cache/>

Parameters

- `cache_timeout` (*int*) – same as `cache_page`; set to 0 for no cache
- `cache_kwargs` (*dict*) – dictionary of kwargs to be passed to `cache_page`
- `initkwargs` – kwargs for `.as_view()`

Returns a view instance

`authentication_classes = [<class 'rest_framework.authentication.SessionAuthentication'>]`

`generator_class`

alias of `OpenAPISchemaGenerator`

`get` (*request*, *version=""*, *format=None*)

`permission_classes = [<class 'rest_framework.permissions.AllowAny'>]`

`public = False`

`renderer_classes = (<class 'drf_yasg.renderers.SwaggerYAMLRenderer'>, <class 'drf_yasg.renderers.JSONRenderer'>)`

`schema = None`

classmethod `with_ui` (*renderer='swagger'*, *cache_timeout=0*, *cache_kwargs=None*)

Instantiate this view with a Web UI renderer, optionally wrapped with `cache_page`. See <https://docs.djangoproject.com/en/1.11/topics/cache/>.

Parameters

- `renderer` (*str*) – UI renderer; allowed values are `swagger`, `redoc`
- `cache_timeout` (*int*) – same as `cache_page`; set to 0 for no cache
- `cache_kwargs` (*dict*) – dictionary of kwargs to be passed to `cache_page`

Returns a view instance

classmethod `without_ui` (*cache_timeout=0*, *cache_kwargs=None*)

Instantiate this view with just JSON and YAML renderers, optionally wrapped with `cache_page`. See <https://docs.djangoproject.com/en/1.11/topics/cache/>.

Parameters

- `cache_timeout` (*int*) – same as `cache_page`; set to 0 for no cache
- `cache_kwargs` (*dict*) – dictionary of kwargs to be passed to `cache_page`

Returns a view instance

c

`drf_yasg.codecs`, 41

e

`drf_yasg.errors`, 43

g

`drf_yasg.generators`, 43

i

`drf_yasg.inspectors`, 46

m

`drf_yasg.middleware`, 55

o

`drf_yasg.openapi`, 55

r

`drf_yasg.renderers`, 63

u

`drf_yasg.utils`, 64

v

`drf_yasg.views`, 67

Symbols

- `_OpenAPICodec` (class in `drf_yasg.codecs`), 41
 - `_Ref` (class in `drf_yasg.openapi`), 60
 - `_SpecRenderer` (class in `drf_yasg.renderers`), 63
 - `_UIRenderer` (class in `drf_yasg.renderers`), 63
 - `_as_odict()` (`drf_yasg.openapi.SwaggerDict` static method), 57
 - `_check_scope()` (`drf_yasg.openapi.ReferenceResolver` method), 62
 - `_dump_dict()` (`drf_yasg.codecs.OpenAPICodecJson` method), 42
 - `_dump_dict()` (`drf_yasg.codecs.OpenAPICodecYaml` method), 42
 - `_dump_dict()` (`drf_yasg.codecs._OpenAPICodec` method), 42
 - `_get_partial_types()` (`drf_yasg.inspectors.FieldInspector` method), 48
 - `_ignore_model_permissions` (`drf_yasg.views.SchemaView` attribute), 67
 - `_insert_extras__()` (`drf_yasg.openapi.SwaggerDict` method), 57
 - `_prepend_inspector_overrides()` (`drf_yasg.inspectors.ViewInspector` method), 50
 - `_validate_flex()` (in module `drf_yasg.codecs`), 41
 - `_validate_swagger_spec_validator()` (in module `drf_yasg.codecs`), 41
- ## A
- `add_manual_parameters()` (`drf_yasg.inspectors.SwaggerAutoSchema` method), 53
 - `apply_cache()` (`drf_yasg.views.SchemaView` class method), 68
 - `as_cached_view()` (`drf_yasg.views.SchemaView` class method), 68
 - `as_odict()` (`drf_yasg.openapi.SwaggerDict` method), 57
 - `authentication_classes` (`drf_yasg.views.SchemaView` attribute), 68
- ## B
- `BaseInspector` (class in `drf_yasg.inspectors`), 46
 - `body_methods` (`drf_yasg.inspectors.ViewInspector` attribute), 50
- ## C
- `CamelCaseJSONFilter` (class in `drf_yasg.inspectors`), 53
 - `charset` (`drf_yasg.renderers._SpecRenderer` attribute), 63
 - `charset` (`drf_yasg.renderers._UIRenderer` attribute), 64
 - `ChoiceFieldInspector` (class in `drf_yasg.inspectors`), 52
 - `codec_class` (`drf_yasg.renderers._SpecRenderer` attribute), 63
 - `codec_class` (`drf_yasg.renderers.OpenAPIRenderer` attribute), 63
 - `codec_class` (`drf_yasg.renderers.SwaggerJSONRenderer` attribute), 63
 - `codec_class` (`drf_yasg.renderers.SwaggerYAMLRenderer` attribute), 63
 - `Contact` (class in `drf_yasg.openapi`), 57
 - `coreapi_field_to_parameter()` (`drf_yasg.inspectors.CoreAPICompatInspector` method), 51
 - `CoreAPICompatInspector` (class in `drf_yasg.inspectors`), 51
 - `create_view()` (`drf_yasg.generators.OpenAPISchemaGenerator` method), 44
- ## D
- `deferred_never_cache()` (in module `drf_yasg.views`), 67
 - `determine_path_prefix()` (`drf_yasg.generators.OpenAPISchemaGenerator` method), 45
 - `DictFieldInspector` (class in `drf_yasg.inspectors`), 52
 - `DjangoRestResponsePagination` (class in `drf_yasg.inspectors`), 51
 - `drf_yasg.codecs` (module), 41
 - `drf_yasg.errors` (module), 43
 - `drf_yasg.generators` (module), 43

drf_yasg.inspectors (module), 46
drf_yasg.middleware (module), 55
drf_yasg.openapi (module), 55
drf_yasg.renderers (module), 63
drf_yasg.utils (module), 64
drf_yasg.views (module), 67

E

encode() (drf_yasg.codecs._OpenAPICodec method), 41
encode_error() (drf_yasg.codecs._OpenAPICodec method), 41
endpoint_enumerator_class (drf_yasg.generators.OpenAPISchemaGenerator attribute), 44
EndpointEnumerator (class in drf_yasg.generators), 43

F

field_inspectors (drf_yasg.inspectors.ViewInspector attribute), 50
field_to_swagger_object() (drf_yasg.inspectors.ChoiceFieldInspector method), 52
field_to_swagger_object() (drf_yasg.inspectors.DictFieldInspector method), 53
field_to_swagger_object() (drf_yasg.inspectors.FieldInspector method), 49
field_to_swagger_object() (drf_yasg.inspectors.FileFieldInspector method), 52
field_to_swagger_object() (drf_yasg.inspectors.InlineSerializerInspector method), 52
field_to_swagger_object() (drf_yasg.inspectors.RelatedFieldInspector method), 52
field_to_swagger_object() (drf_yasg.inspectors.SimpleFieldInspector method), 52
field_to_swagger_object() (drf_yasg.inspectors.StringDefaultFieldInspector method), 53
FieldInspector (class in drf_yasg.inspectors), 48
FileFieldInspector (class in drf_yasg.inspectors), 52
filter_inspectors (drf_yasg.inspectors.ViewInspector attribute), 50
filter_none() (in module drf_yasg.utils), 66
FilterInspector (class in drf_yasg.inspectors), 47
force_serializer_instance() (in module drf_yasg.utils), 66
format (drf_yasg.renderers.OpenAPIRenderer attribute), 63
format (drf_yasg.renderers.ReDocAlphaRenderer attribute), 64

format (drf_yasg.renderers.ReDocRenderer attribute), 64
format (drf_yasg.renderers.SwaggerJSONRenderer attribute), 63
format (drf_yasg.renderers.SwaggerUIRenderer attribute), 64
format (drf_yasg.renderers.SwaggerYAMLRenderer attribute), 63
FORMAT_BASE64 (in module drf_yasg.openapi), 56
FORMAT_BINARY (in module drf_yasg.openapi), 56
FORMAT_DATE (in module drf_yasg.openapi), 56
FORMAT_DATETIME (in module drf_yasg.openapi), 56
FORMAT_DOUBLE (in module drf_yasg.openapi), 56
FORMAT_EMAIL (in module drf_yasg.openapi), 56
FORMAT_FLOAT (in module drf_yasg.openapi), 56
FORMAT_INT32 (in module drf_yasg.openapi), 56
FORMAT_INT64 (in module drf_yasg.openapi), 56
FORMAT_IPV4 (in module drf_yasg.openapi), 56
FORMAT_IPV6 (in module drf_yasg.openapi), 56
FORMAT_PASSWORD (in module drf_yasg.openapi), 56
FORMAT_SLUG (in module drf_yasg.openapi), 56
FORMAT_URI (in module drf_yasg.openapi), 56
FORMAT_UUID (in module drf_yasg.openapi), 56

G

generate_swagger_object() (drf_yasg.codecs._OpenAPICodec method), 42
generator_class (drf_yasg.views.SchemaView attribute), 68
get() (drf_yasg.openapi.ReferenceResolver method), 62
get() (drf_yasg.views.SchemaView method), 68
get_api_endpoints() (drf_yasg.generators.EndpointEnumerator method), 43
get_auth_urls() (drf_yasg.renderers._UIRenderer method), 64
get_base_path() (drf_yasg.openapi.Swagger class method), 58
get_consumes() (drf_yasg.inspectors.SwaggerAutoSchema method), 53
get_consumes() (in module drf_yasg.utils), 67
get_default_responses() (drf_yasg.inspectors.SwaggerAutoSchema method), 53
get_description() (drf_yasg.inspectors.SwaggerAutoSchema method), 53
get_endpoints() (drf_yasg.generators.OpenAPISchemaGenerator method), 44
get_filter_parameters() (drf_yasg.inspectors.CoreAPICompatInspector method), 51
get_filter_parameters() (drf_yasg.inspectors.FilterInspector method), 48
get_filter_parameters() (drf_yasg.inspectors.ViewInspector method), 50
get_operation() (drf_yasg.generators.OpenAPISchemaGenerator method), 45

- [get_operation\(\) \(drf_yasg.inspectors.SwaggerAutoSchema method\), 53](#)
[get_operation\(\) \(drf_yasg.inspectors.ViewInspector method\), 50](#)
[get_operation_id\(\) \(drf_yasg.inspectors.SwaggerAutoSchema method\), 53](#)
[get_operation_keys\(\) \(drf_yasg.generators.OpenAPISchemaGenerator method\), 44](#)
[get_overrides\(\) \(drf_yasg.generators.OpenAPISchemaGenerator method\), 46](#)
[get_paginated_response\(\) \(drf_yasg.inspectors.DjangoRestResponsePagination method\), 52](#)
[get_paginated_response\(\) \(drf_yasg.inspectors.PaginatorInspector method\), 48](#)
[get_paginated_response\(\) \(drf_yasg.inspectors.ViewInspector method\), 50](#)
[get_pagination_parameters\(\) \(drf_yasg.inspectors.ViewInspector method\), 50](#)
[get_paginator_parameters\(\) \(drf_yasg.inspectors.CoreAPICompatInspector method\), 51](#)
[get_paginator_parameters\(\) \(drf_yasg.inspectors.PaginatorInspector method\), 48](#)
[get_parameter_name\(\) \(drf_yasg.inspectors.InlineSerializerInspector method\), 52](#)
[get_path_from_regex\(\) \(drf_yasg.generators.EndpointEnumerator method\), 43](#)
[get_path_item\(\) \(drf_yasg.generators.OpenAPISchemaGenerator method\), 46](#)
[get_path_parameters\(\) \(drf_yasg.generators.OpenAPISchemaGenerator method\), 46](#)
[get_paths\(\) \(drf_yasg.generators.OpenAPISchemaGenerator method\), 45](#)
[get_produces\(\) \(drf_yasg.inspectors.SwaggerAutoSchema method\), 54](#)
[get_produces\(\) \(in module drf_yasg.utils\), 67](#)
[get_property_name\(\) \(drf_yasg.inspectors.InlineSerializerInspector method\), 52](#)
[get_query_parameters\(\) \(drf_yasg.inspectors.SwaggerAutoSchema method\), 54](#)
[get_query_serializer\(\) \(drf_yasg.inspectors.SwaggerAutoSchema method\), 54](#)
[get_redoc_settings\(\) \(drf_yasg.renderers._UIRenderer method\), 64](#)
[get_request_body_parameters\(\) \(drf_yasg.inspectors.SwaggerAutoSchema method\), 54](#)
[get_request_body_schema\(\) \(drf_yasg.inspectors.SwaggerAutoSchema method\), 54](#)
[get_request_form_parameters\(\) \(drf_yasg.inspectors.SwaggerAutoSchema method\), 54](#)
[get_request_parameters\(\) \(drf_yasg.inspectors.InlineSerializerInspector method\), 52](#)
[get_request_parameters\(\) \(drf_yasg.inspectors.SerializerInspector method\), 49](#)
[get_request_serializer\(\) \(drf_yasg.inspectors.SwaggerAutoSchema method\), 54](#)
[get_response_schemas\(\) \(drf_yasg.inspectors.SwaggerAutoSchema method\), 54](#)
[get_response_serializers\(\) \(drf_yasg.inspectors.SwaggerAutoSchema method\), 54](#)
[get_responses\(\) \(drf_yasg.inspectors.SwaggerAutoSchema method\), 55](#)
[get_schema\(\) \(drf_yasg.generators.OpenAPISchemaGenerator method\), 44](#)
[get_schema\(\) \(drf_yasg.inspectors.InlineSerializerInspector method\), 52](#)
[get_schema\(\) \(drf_yasg.inspectors.SerializerInspector method\), 50](#)
[get_schema_view\(\) \(in module drf_yasg.views\), 67](#)
[get_security\(\) \(drf_yasg.inspectors.SwaggerAutoSchema method\), 55](#)
[get_swagger_ui_settings\(\) \(drf_yasg.renderers._UIRenderer method\), 64](#)
[get_tags\(\) \(drf_yasg.inspectors.SwaggerAutoSchema method\), 55](#)
[get_view_serializer\(\) \(drf_yasg.inspectors.SwaggerAutoSchema method\), 55](#)
[getdefault\(\) \(drf_yasg.openapi.ReferenceResolver method\), 62](#)
[guess_response_status\(\) \(in module drf_yasg.utils\), 66](#)
- ## H
- [has\(\) \(drf_yasg.openapi.ReferenceResolver method\), 62](#)
- ## I
- [IN_BODY \(in module drf_yasg.openapi\), 56](#)
[IN_FORM \(in module drf_yasg.openapi\), 56](#)
[IN_HEADER \(in module drf_yasg.openapi\), 56](#)
[IN_PATH \(in module drf_yasg.openapi\), 56](#)
[IN_QUERY \(in module drf_yasg.openapi\), 56](#)
[Info \(class in drf_yasg.openapi\), 57](#)
[InlineSerializerInspector \(class in drf_yasg.inspectors\), 52](#)
[is_camel_case\(\) \(drf_yasg.inspectors.CamelCaseJSONFilter method\), 53](#)
[is_list_view\(\) \(in module drf_yasg.utils\), 66](#)

Items (class in drf_yasg.openapi), 59

K

keys() (drf_yasg.openapi.ReferenceResolver method), 63

L

License (class in drf_yasg.openapi), 57

M

make_body_parameter() (drf_yasg.inspectors.SwaggerAutoSchema method), 55

make_swagger_name() (in module drf_yasg.openapi), 56

media_type (drf_yasg.codecs._OpenAPICodec attribute), 41

media_type (drf_yasg.codecs.OpenAPICodecJson attribute), 42

media_type (drf_yasg.codecs.OpenAPICodecYaml attribute), 42

media_type (drf_yasg.renderers._UIRenderer attribute), 64

media_type (drf_yasg.renderers.OpenAPIRenderer attribute), 63

media_type (drf_yasg.renderers.SwaggerJSONRenderer attribute), 63

media_type (drf_yasg.renderers.SwaggerYAMLRenderer attribute), 63

N

no_body (in module drf_yasg.utils), 64

NotHandled (in module drf_yasg.inspectors), 46

O

OpenAPICodecJson (class in drf_yasg.codecs), 42

OpenAPICodecYaml (class in drf_yasg.codecs), 42

OpenAPIRenderer (class in drf_yasg.renderers), 63

OpenAPISchemaGenerator (class in drf_yasg.generators), 43

Operation (class in drf_yasg.openapi), 59

OR_REF (drf_yasg.openapi.Schema attribute), 60

P

paginator_inspectors (drf_yasg.inspectors.ViewInspector attribute), 50

PaginatorInspector (class in drf_yasg.inspectors), 48

param_list_to_odict() (in module drf_yasg.utils), 66

Parameter (class in drf_yasg.openapi), 59

PathItem (class in drf_yasg.openapi), 58

Paths (class in drf_yasg.openapi), 58

permission_classes (drf_yasg.views.SchemaView attribute), 68

probe_field_inspectors() (drf_yasg.inspectors.FieldInspector method), 49

probe_inspectors() (drf_yasg.inspectors.BaseInspector method), 47

process_exception() (drf_yasg.middleware.SwaggerExceptionMiddleware method), 55

process_result() (drf_yasg.inspectors.BaseInspector method), 47

process_result() (drf_yasg.inspectors.CamelCaseJSONFilter method), 53

public (drf_yasg.views.SchemaView attribute), 68

R

ReDocAlphaRenderer (class in drf_yasg.renderers), 64

ReDocRenderer (class in drf_yasg.renderers), 64

ref_name_re (drf_yasg.openapi._Ref attribute), 61

ReferenceResolver (class in drf_yasg.openapi), 61

ReferencingSerializerInspector (class in drf_yasg.inspectors), 52

RelatedFieldInspector (class in drf_yasg.inspectors), 52

render() (drf_yasg.renderers._SpecRenderer method), 63

render() (drf_yasg.renderers._UIRenderer method), 64

renderer_classes (drf_yasg.views.SchemaView attribute), 68

replace_version() (drf_yasg.generators.EndpointEnumerator method), 43

resolve() (drf_yasg.openapi._Ref method), 61

resolve_ref() (in module drf_yasg.openapi), 61

Response (class in drf_yasg.openapi), 61

Responses (class in drf_yasg.openapi), 61

S

Schema (class in drf_yasg.openapi), 60

schema (drf_yasg.views.SchemaView attribute), 68

SCHEMA_DEFINITIONS (in module drf_yasg.openapi), 56

SchemaRef (class in drf_yasg.openapi), 61

SchemaView (class in drf_yasg.views), 67

scopes (drf_yasg.openapi.ReferenceResolver attribute), 63

serializer_to_parameters() (drf_yasg.inspectors.ViewInspector method), 50

serializer_to_schema() (drf_yasg.inspectors.ViewInspector method), 51

SerializerInspector (class in drf_yasg.inspectors), 49

set() (drf_yasg.openapi.ReferenceResolver method), 62

set_context() (drf_yasg.renderers._UIRenderer method), 64

setdefault() (drf_yasg.openapi.ReferenceResolver method), 62

should_filter() (drf_yasg.inspectors.ViewInspector method), 51

should_include_endpoint() (drf_yasg.generators.EndpointEnumerator method), 43

should_page() (drf_yasg.inspectors.ViewInspector method), 51

SimpleFieldInspector (class in drf_yasg.inspectors), 52

StringDefaultFieldInspector (class in drf_yasg.inspectors), 53

Swagger (class in drf_yasg.openapi), 57

swagger_auto_schema() (in module drf_yasg.utils), 64

SwaggerAutoSchema (class in drf_yasg.inspectors), 53

SwaggerDict (class in drf_yasg.openapi), 56

SwaggerError, 43

SwaggerExceptionMiddleware (class in drf_yasg.middleware), 55

SwaggerGenerationError, 43

SwaggerJSONRenderer (class in drf_yasg.renderers), 63

SwaggerUIRenderer (class in drf_yasg.renderers), 64

SwaggerValidationError, 43

SwaggerYAMLRenderer (class in drf_yasg.renderers), 63

T

template (drf_yasg.renderers._UIRenderer attribute), 64

template (drf_yasg.renderers.ReDocAlphaRenderer attribute), 64

template (drf_yasg.renderers.ReDocRenderer attribute), 64

template (drf_yasg.renderers.SwaggerUIRenderer attribute), 64

TYPE_ARRAY (in module drf_yasg.openapi), 56

TYPE_BOOLEAN (in module drf_yasg.openapi), 56

TYPE_FILE (in module drf_yasg.openapi), 56

TYPE_INTEGER (in module drf_yasg.openapi), 56

TYPE_NUMBER (in module drf_yasg.openapi), 55

TYPE_OBJECT (in module drf_yasg.openapi), 55

TYPE_STRING (in module drf_yasg.openapi), 55

U

unescape() (drf_yasg.generators.EndpointEnumerator method), 43

unescape_path() (drf_yasg.generators.EndpointEnumerator method), 43

url (drf_yasg.generators.OpenAPISchemaGenerator attribute), 44

use_definitions (drf_yasg.inspectors.InlineSerializerInspector attribute), 52

use_definitions (drf_yasg.inspectors.ReferencingSerializerInspector attribute), 52

V

validators (drf_yasg.codecs._OpenAPICodec attribute), 41

validators (drf_yasg.renderers._SpecRenderer attribute), 63

VALIDATORS (in module drf_yasg.codecs), 41

ViewInspector (class in drf_yasg.inspectors), 50

W

with_scope() (drf_yasg.openapi.ReferenceResolver method), 62

with_ui() (drf_yasg.views.SchemaView class method), 68

with_validators() (drf_yasg.renderers._SpecRenderer class method), 63

without_ui() (drf_yasg.views.SchemaView class method), 68

Y

yaml_sane_dump() (in module drf_yasg.codecs), 42

yaml_sane_load() (in module drf_yasg.codecs), 42