
flask_api

Rubén Rodríguez Ramírez

Dec 28, 2021

CONTENTS

1	Installation	3
1.1	1. Linux packages	3
1.2	2. RabbitMQ configuration	3
1.3	3. Python dependencies	3
1.4	4. Domain configuration	4
1.5	5. Environment configuration	4
1.6	6. uWSGI configuration	4
1.7	7. Nginx configuration	4
1.8	8. Supervisor configuration	4
1.9	9. Log directories	5
1.10	10. Supervisor systemd unit file	5
1.11	How to usage	6
1.12	Optional installation	6
2	Skeleton app structure	7
2.1	app	8
2.2	database	348
2.3	tests	377
2.4	config	399
3	Flask command line	437
4	Changelog	439
4.1	2.0.6 (2021-12-28)	439
4.2	Bug Fixes	439
4.3	2.0.5 (2021-12-27)	439
4.4	Bug Fixes	439
4.5	Bug Fixes	439
4.6	Build System	439
4.7	Bug Fixes	440
4.8	Code Refactoring	440
4.9	Bug Fixes	440
4.10	Build System	440
4.11	Code Refactoring	440
4.12	Bug Fixes	441
4.13	Build System	441
4.14	Code Refactoring	441
4.15	BREAKING CHANGES	442
4.16	Features	442
4.17	Code Refactoring	442

4.18	Build System	442
4.19	Bug Fixes	442
4.20	Features	442
4.21	Features	443
4.22	BREAKING CHANGES	443
4.23	build	443
4.24	Features	443
4.25	Refactor	443
4.26	Docs	443
4.27	BREAKING CHANGES	444
4.28	Features	444
4.29	Build	444
4.30	Features	444
4.31	Features	444
4.32	BREAKING CHANGES	445
4.33	Features	445
4.34	Bug Fixes	445
4.35	Build	445
5	Note	447
6	Indices and tables	449
	Python Module Index	451
	Index	453

Flask-api is a small API project for creating users and files (Microsoft Word and PDF). These files contain data about users registered in the project.

The project is developed in Python 3.7 and use next main libraries:

- : microframework.
- : SQL database engine.
- : simple and small ORM.
- : asynchronous task queue/job.
- : message broker.
- : web server, reverse proxy, etc.
- : Web Server Gateway Interface (WSGI) server implementation.
- : monitoring and administrating Celery clusters.
- : client/server system that allows its users to monitor and control a number of processes on UNIX-like operating systems.

INSTALLATION

1.1 1. Linux packages

These packages are required for the project installation:

```
sudo add-apt-repository ppa:deadsnakes/ppa
sudo apt-get update
sudo apt-get install autoconf build-essential cmake libcap-dev libffi-dev libpcre3-dev
↳ librabbitmq-dev libreoffice-writer libtool libxml2-dev libxslt1-dev libxslt1.1 pkg-
↳ config magic nginx python3-distutils python3.7 python3.7-dev python3.7-venv rabbitmq-
↳ server uuid-dev uwsgi uwsgi-src
sudo reboot
```

1.2 2. RabbitMQ configuration

Required plugins for monitoring our brokers in Flower:

```
sudo rabbitmq-plugins enable rabbitmq_management
sudo service rabbitmq-server restart
```

1.3 3. Python dependencies

Install Python dependencies:

```
python3.7 -m venv venv
source venv/bin/activate
pip install -r requirements.txt --no-cache-dir
```

1.4 4. Domain configuration

Add local domain to our */etc/hosts* file:

```
127.0.0.1 flask-api.prod
```

1.5 5. Environment configuration

Create a new **.env** file based on *.env.example* file.

1.6 6. uWSGI configuration

Create a new **uwsgi.ini** file based on *uwsgi.ini.example*.

username and *project_path* must to be filled with appropriate values.

www-data group must to be added to your user:

```
sudo usermod -a -G www-data username
```

1.7 7. Nginx configuration

Create a new **flask_api** file based on *docs/examples/flask_api.nginx.example* file.

Replace *uwsgi_pass* variable with the value in *socket* variable from **uwsgi.ini** file.

Move **flask_api** file to */etc/nginx/sites-available* directory:

```
sudo mv docs/examples/flask_api /etc/nginx/sites-available
sudo ln -s /etc/nginx/sites-available/flask_api /etc/nginx/sites-enabled/flask_api
sudo systemctl restart nginx
```

1.8 8. Supervisor configuration

1.8.1 8.1 Main configuration

Create a new **supervisord.conf** file based on *docs/examples/supervisor/supervisord.conf.example* file in the root project.

command, *directory* and *username* variables must to be filled with appropriate values. These variables are below *Mr Developer* comment.

1.8.2 8.2 Other configurations

Create a new directory named *supervisor* in the root path and create next files based on *docs/examples/supervisor* example files:

1. celery.conf
2. flower.conf
3. uwsgi.conf

username and *path* variables must to be replaced with appropriate values.

1.9 9. Log directories

Create next log directories:

1. log/app
2. log/celery
3. log/flower
4. log/uwsgi

1.10 10. Supervisor systemd unit file

Create a new **flask_api_supervisor.service** file based on *docs/examples/flask_api_supervisor.service.example* file.

username, *usergroup* and *path* variables must to be filled with appropriate values.

Move file to */etc/systemd/system* directory and we run next commands:

```
sudo systemctl enable flask_api_supervisor.service
sudo systemctl daemon-reload
sudo systemctl start flask_api_supervisor.service
```

The systemd unit file start up the project if the system is reboot or shutdown.

For checking process status in command line:

```
sudo systemctl status flask_api_supervisor.service
```

For restart all processes in command line:

```
sudo systemctl restart flask_api_supervisor.service
```

This command reread the supervisor configuration files, stop all processes and start them again.

1.11 How to usage

The setup is finished, we only need to create the database tables and fill them with fake data. We open a terminal in the root project and run next commands:

```
./venv/bin/flask init-db
./venv/bin/flask migrate
./venv/bin/flask seed
```

You can use an API client such as Insomnia or Postman and starting to consume the API!

You can see the processes status here: <http://flask-api.prod/supervisor>

The credentials are user:123 by default, you can change the credentials in supervisord.conf file in *inet_http_server* section.

You can management the Celery tasks status here: <http://flask-api.prod:5555/flower/>

1.12 Optional installation

This project use for logging configuration. The config file is already defined you only need to do these steps:

1. Create new **flask_api.logrotate** file based on *docs/examples/flask_api.logrotate.example*.
2. *path*, *username* and *usergroup* variables must to be filled with appropriate values.
3. Move flask_api_logrotate to */etc/logrotate.d*:

```
sudo mv docs/examples/flask_api.logrotate /etc/logrotate.d
```

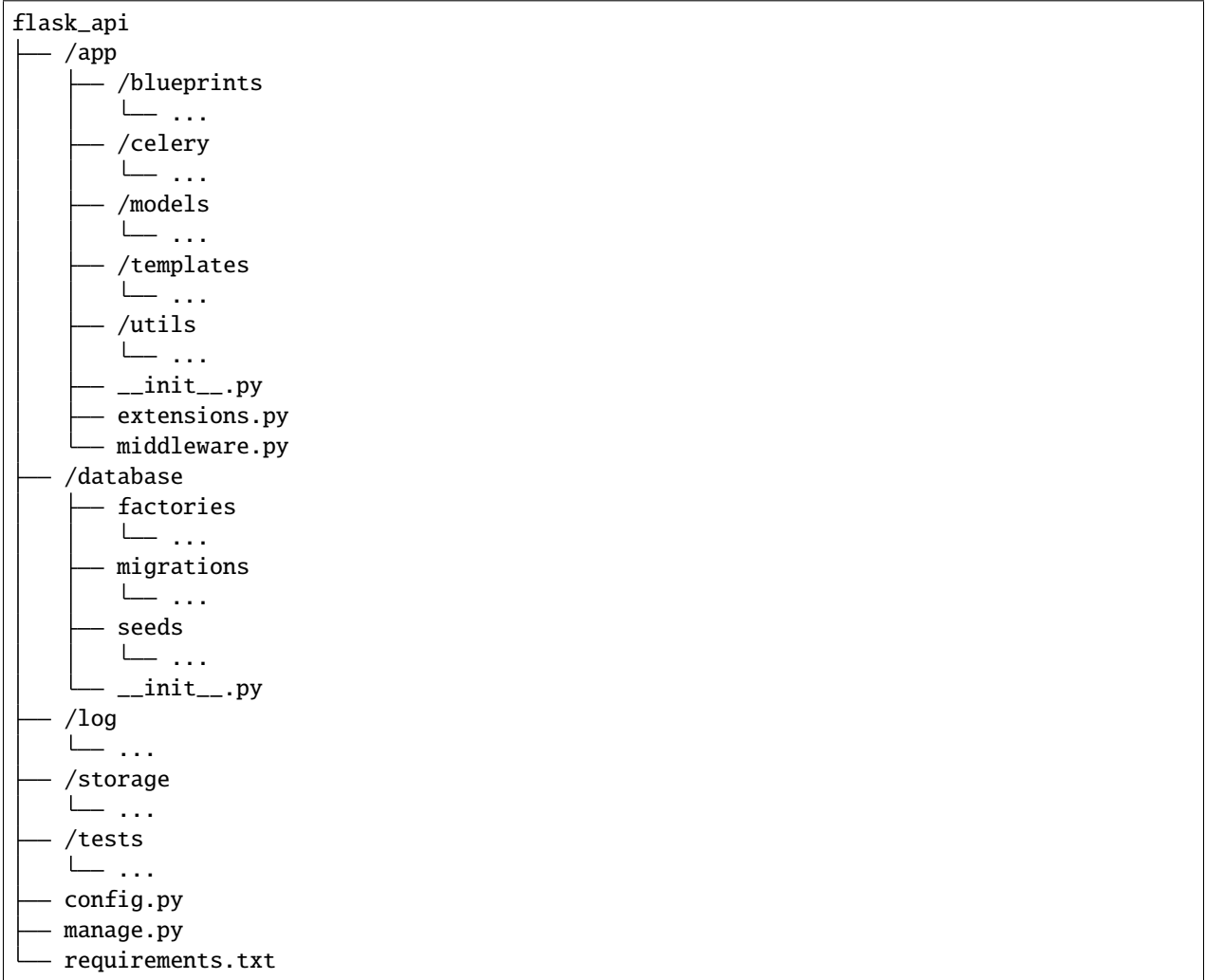
4. Restart logrotate service:

```
sudo service log rotate restart
```

A new log file will be created every day.

SKELETON APP STRUCTURE

The project structure looks like this:



<i>app</i>	Package for building a Flask application.
<i>database</i>	Package for managing the database.
<i>tests</i>	Package for testing the application.
<i>config</i>	Module loads the application's configuration.

2.1 app

Description

Package for building a Flask application.

The app package loads application configuration and registers middleware, blueprints, database models, etc.

Modules

<i>app.blueprints</i>	Registers Flask blueprints.
<i>app.celery</i>	Runs Celery and registers Celery tasks.
<i>app.exceptions</i>	Module for managing exceptions.
<i>app.extensions</i>	Registers third party extensions.
<i>app.managers</i>	Registers database managers.
<i>app.middleware</i>	WSGI middleware for validating requests content type.
<i>app.models</i>	Registers database models.
<i>app.serializers</i>	Modules for managing data from requests and responses.
<i>app.services</i>	Registers services for managing business logic.
<i>app.swagger</i>	Models registered in Swagger.
<i>app.utils</i>	Collection of functions and classes which make common patterns shorter and easier.

2.1.1 app.blueprints

Description

Registers Flask blueprints.

Modules

<i>app.blueprints.auth</i>
<i>app.blueprints.base</i>
<i>app.blueprints.documents</i>
<i>app.blueprints.roles</i>
<i>app.blueprints.tasks</i>
<i>app.blueprints.users</i>

app.blueprints.auth**Description****Classes**

AuthBaseResource([api])

AuthUserLoginResource([api])

AuthUserLogoutResource([api])

RequestResetPasswordResource([api])

ResetPasswordResource([api])

app.blueprints.auth.AuthBaseResource

class app.blueprints.auth.**AuthBaseResource**(api=None, *args, **kwargs)
Bases: flask_restx.resource.Resource

Attributes

AuthBaseResource.auth_service

AuthBaseResource.decorators The canonical way to decorate class-based views is to decorate the return value of `as_view()`.

AuthBaseResource.method_decorators

AuthBaseResource.methods A list of methods this view can handle.

AuthBaseResource.provide_automatic_options Setting this disables or force-enables the automatic options handling.

AuthBaseResource.representations

app.blueprints.auth.AuthBaseResource.auth_service

`AuthBaseResource.auth_service` = <app.services.auth.AuthService object>

app.blueprints.auth.AuthBaseResource.decorators

AuthBaseResource.decorators: `List[Callable] = []`

The canonical way to decorate class-based views is to decorate the return value of `as_view()`. However since this moves parts of the logic from the class declaration to the place where it's hooked into the routing system.

You can place one or more decorators in this list and whenever the view function is created the result is automatically decorated.

New in version 0.8.

app.blueprints.auth.AuthBaseResource.method_decorators

AuthBaseResource.method_decorators = `[]`

app.blueprints.auth.AuthBaseResource.methods

AuthBaseResource.methods: `Optional[List[str]] = None`

A list of methods this view can handle.

app.blueprints.auth.AuthBaseResource.provide_automatic_options

AuthBaseResource.provide_automatic_options: `Optional[bool] = None`

Setting this disables or force-enables the automatic options handling.

app.blueprints.auth.AuthBaseResource.representations

AuthBaseResource.representations = `None`

Methods

<code>AuthBaseResource.__init__([api])</code>	
<hr/>	
<code>AuthBaseResource.as_view(name, *class_args, ...)</code>	Converts the class into an actual view function that can be used with the routing system.
<code>AuthBaseResource.dispatch_request(*args, ...)</code>	Subclasses have to override this method to implement the actual view function code.
<code>AuthBaseResource.validate_payload(func)</code>	Perform a payload validation on expected model if necessary

app.blueprints.auth.AuthBaseResource.__init__

`AuthBaseResource.__init__(api=None, *args, **kwargs)`

app.blueprints.auth.AuthBaseResource.as_view

classmethod `AuthBaseResource.as_view(name: str, *class_args: Any, **class_kwargs: Any) → Callable`

Converts the class into an actual view function that can be used with the routing system. Internally this generates a function on the fly which will instantiate the `View` on each request and call the `dispatch_request()` method on it.

The arguments passed to `as_view()` are forwarded to the constructor of the class.

app.blueprints.auth.AuthBaseResource.dispatch_request

`AuthBaseResource.dispatch_request(*args, **kwargs)`

Subclasses have to override this method to implement the actual view function code. This method is called with all the arguments from the URL rule.

app.blueprints.auth.AuthBaseResource.validate_payload

`AuthBaseResource.validate_payload(func)`

Perform a payload validation on expected model if necessary

app.blueprints.auth.AuthUserLoginResource

class `app.blueprints.auth.AuthUserLoginResource(api=None, *args, **kwargs)`

Bases: `app.blueprints.auth.AuthBaseResource`

Attributes

<code>AuthUserLoginResource.auth_service</code>	
<code>AuthUserLoginResource.decorators</code>	The canonical way to decorate class-based views is to decorate the return value of <code>as_view()</code> .
<code>AuthUserLoginResource.method_decorators</code>	
<code>AuthUserLoginResource.methods</code>	A list of methods this view can handle.
<code>AuthUserLoginResource.provide_automatic_options</code>	Setting this disables or force-enables the automatic options handling.
<code>AuthUserLoginResource.representations</code>	

app.blueprints.auth.AuthUserLoginResource.auth_service

```
AuthUserLoginResource.auth_service = <app.services.auth.AuthService object>
```

app.blueprints.auth.AuthUserLoginResource.decorators

```
AuthUserLoginResource.decorators: List[Callable] = []
```

The canonical way to decorate class-based views is to decorate the return value of `as_view()`. However since this moves parts of the logic from the class declaration to the place where it's hooked into the routing system.

You can place one or more decorators in this list and whenever the view function is created the result is automatically decorated.

New in version 0.8.

app.blueprints.auth.AuthUserLoginResource.method_decorators

```
AuthUserLoginResource.method_decorators = []
```

app.blueprints.auth.AuthUserLoginResource.methods

```
AuthUserLoginResource.methods: Optional[List[str]] = {'POST'}
```

A list of methods this view can handle.

app.blueprints.auth.AuthUserLoginResource.provide_automatic_options

```
AuthUserLoginResource.provide_automatic_options: Optional[bool] = None
```

Setting this disables or force-enables the automatic options handling.

app.blueprints.auth.AuthUserLoginResource.representations

```
AuthUserLoginResource.representations = None
```

Methods

<code>AuthUserLoginResource.__init__([api])</code>	
<hr/>	
<code>AuthUserLoginResource.as_view(name, ...)</code>	Converts the class into an actual view function that can be used with the routing system.
<code>AuthUserLoginResource.dispatch_request(...)</code>	Subclasses have to override this method to implement the actual view function code.
<hr/>	
<code>AuthUserLoginResource.post()</code>	
<hr/>	
<code>AuthUserLoginResource.validate_payload(func)</code>	Perform a payload validation on expected model if necessary

app.blueprints.auth.AuthUserLoginResource.__init__

`AuthUserLoginResource.__init__(api=None, *args, **kwargs)`

app.blueprints.auth.AuthUserLoginResource.as_view

classmethod `AuthUserLoginResource.as_view(name: str, *class_args: Any, **class_kwargs: Any) → Callable`

Converts the class into an actual view function that can be used with the routing system. Internally this generates a function on the fly which will instantiate the `View` on each request and call the `dispatch_request()` method on it.

The arguments passed to `as_view()` are forwarded to the constructor of the class.

app.blueprints.auth.AuthUserLoginResource.dispatch_request

`AuthUserLoginResource.dispatch_request(*args, **kwargs)`

Subclasses have to override this method to implement the actual view function code. This method is called with all the arguments from the URL rule.

app.blueprints.auth.AuthUserLoginResource.post

`AuthUserLoginResource.post() → tuple`

app.blueprints.auth.AuthUserLoginResource.validate_payload

`AuthUserLoginResource.validate_payload(func)`

Perform a payload validation on expected model if necessary

app.blueprints.auth.AuthUserLogoutResource

class `app.blueprints.auth.AuthUserLogoutResource(api=None, *args, **kwargs)`

Bases: `app.blueprints.auth.AuthBaseResource`

Attributes

<code>AuthUserLogoutResource.auth_service</code>	
<code>AuthUserLogoutResource.decorators</code>	The canonical way to decorate class-based views is to decorate the return value of <code>as_view()</code> .
<code>AuthUserLogoutResource.method_decorators</code>	
<code>AuthUserLogoutResource.methods</code>	A list of methods this view can handle.
<code>AuthUserLogoutResource.provide_automatic_options</code>	Setting this disables or force-enables the automatic options handling.

continues on next page

Table 9 – continued from previous page

*AuthUserLogoutResource.
representations*

app.blueprints.auth.AuthUserLogoutResource.auth_service

`AuthUserLogoutResource.auth_service = <app.services.auth.AuthService object>`

app.blueprints.auth.AuthUserLogoutResource.decorators

`AuthUserLogoutResource.decorators: List[Callable] = []`

The canonical way to decorate class-based views is to decorate the return value of `as_view()`. However since this moves parts of the logic from the class declaration to the place where it's hooked into the routing system.

You can place one or more decorators in this list and whenever the view function is created the result is automatically decorated.

New in version 0.8.

app.blueprints.auth.AuthUserLogoutResource.method_decorators

`AuthUserLogoutResource.method_decorators = []`

app.blueprints.auth.AuthUserLogoutResource.methods

`AuthUserLogoutResource.methods: Optional[List[str]] = {'POST'}`

A list of methods this view can handle.

app.blueprints.auth.AuthUserLogoutResource.provide_automatic_options

`AuthUserLogoutResource.provide_automatic_options: Optional[bool] = None`

Setting this disables or force-enables the automatic options handling.

app.blueprints.auth.AuthUserLogoutResource.representations

`AuthUserLogoutResource.representations = None`

Methods

*AuthUserLogoutResource.
__init__([api])*

<i>AuthUserLogoutResource.as_view(name, ...)</i>	Converts the class into an actual view function that can be used with the routing system.
--	---

<i>AuthUserLogoutResource.dispatch_request(...)</i>	Subclasses have to override this method to implement the actual view function code.
---	---

continues on next page

Table 10 – continued from previous page

<i>AuthUserLogoutResource.post()</i>	
<i>AuthUserLogoutResource.validate_payload(func)</i>	Perform a payload validation on expected model if necessary

app.blueprints.auth.AuthUserLogoutResource.__init__

`AuthUserLogoutResource.__init__(api=None, *args, **kwargs)`

app.blueprints.auth.AuthUserLogoutResource.as_view

classmethod `AuthUserLogoutResource.as_view(name: str, *class_args: Any, **class_kwargs: Any) → Callable`

Converts the class into an actual view function that can be used with the routing system. Internally this generates a function on the fly which will instantiate the `View` on each request and call the `dispatch_request()` method on it.

The arguments passed to `as_view()` are forwarded to the constructor of the class.

app.blueprints.auth.AuthUserLogoutResource.dispatch_request

`AuthUserLogoutResource.dispatch_request(*args, **kwargs)`

Subclasses have to override this method to implement the actual view function code. This method is called with all the arguments from the URL rule.

app.blueprints.auth.AuthUserLogoutResource.post

`AuthUserLogoutResource.post() → tuple`

app.blueprints.auth.AuthUserLogoutResource.validate_payload

`AuthUserLogoutResource.validate_payload(func)`

Perform a payload validation on expected model if necessary

app.blueprints.auth.RequestResetPasswordResource

class `app.blueprints.auth.RequestResetPasswordResource(api=None, *args, **kwargs)`

Bases: `app.blueprints.auth.AuthBaseResource`

Attributes

<i>RequestResetPasswordResource.auth_service</i>	
<i>RequestResetPasswordResource.decorators</i>	The canonical way to decorate class-based views is to decorate the return value of <code>as_view()</code> .
<i>RequestResetPasswordResource.method_decorators</i>	
<i>RequestResetPasswordResource.methods</i>	A list of methods this view can handle.
<i>RequestResetPasswordResource.provide_automatic_options</i>	Setting this disables or force-enables the automatic options handling.
<i>RequestResetPasswordResource.representations</i>	

app.blueprints.auth.RequestResetPasswordResource.auth_service

`RequestResetPasswordResource.auth_service = <app.services.auth.AuthService object>`

app.blueprints.auth.RequestResetPasswordResource.decorators

`RequestResetPasswordResource.decorators: List[Callable] = []`

The canonical way to decorate class-based views is to decorate the return value of `as_view()`. However since this moves parts of the logic from the class declaration to the place where it's hooked into the routing system.

You can place one or more decorators in this list and whenever the view function is created the result is automatically decorated.

New in version 0.8.

app.blueprints.auth.RequestResetPasswordResource.method_decorators

`RequestResetPasswordResource.method_decorators = []`

app.blueprints.auth.RequestResetPasswordResource.methods

`RequestResetPasswordResource.methods: Optional[List[str]] = {'POST'}`

A list of methods this view can handle.

app.blueprints.auth.RequestResetPasswordResource.provide_automatic_options

`RequestResetPasswordResource.provide_automatic_options: Optional[bool] = None`

Setting this disables or force-enables the automatic options handling.

app.blueprints.auth.RequestResetPasswordResource.representations

`RequestResetPasswordResource.representations = None`

Methods

<code>RequestResetPasswordResource.__init__([api])</code>	
<code>RequestResetPasswordResource.as_view(name, ...)</code>	Converts the class into an actual view function that can be used with the routing system.
<code>RequestResetPasswordResource.dispatch_request(...)</code>	Subclasses have to override this method to implement the actual view function code.
<code>RequestResetPasswordResource.post()</code>	
<code>RequestResetPasswordResource.validate_payload(func)</code>	Perform a payload validation on expected model if necessary

app.blueprints.auth.RequestResetPasswordResource.__init__

`RequestResetPasswordResource.__init__(api=None, *args, **kwargs)`

app.blueprints.auth.RequestResetPasswordResource.as_view

classmethod `RequestResetPasswordResource.as_view(name: str, *class_args: Any, **class_kwargs: Any) → Callable`

Converts the class into an actual view function that can be used with the routing system. Internally this generates a function on the fly which will instantiate the `View` on each request and call the `dispatch_request()` method on it.

The arguments passed to `as_view()` are forwarded to the constructor of the class.

app.blueprints.auth.RequestResetPasswordResource.dispatch_request

`RequestResetPasswordResource.dispatch_request(*args, **kwargs)`

Subclasses have to override this method to implement the actual view function code. This method is called with all the arguments from the URL rule.

app.blueprints.auth.RequestResetPasswordResource.post

`RequestResetPasswordResource.post()` → tuple

app.blueprints.auth.RequestResetPasswordResource.validate_payload

`RequestResetPasswordResource.validate_payload(func)`
Perform a payload validation on expected model if necessary

app.blueprints.auth.ResetPasswordResource

class `app.blueprints.auth.ResetPasswordResource`(*api=None, *args, **kwargs*)
Bases: `app.blueprints.auth.AuthBaseResource`

Attributes

<hr/> <i>ResetPasswordResource.auth_service</i>	
<i>ResetPasswordResource.decorators</i>	The canonical way to decorate class-based views is to decorate the return value of <code>as_view()</code> .
<hr/> <i>ResetPasswordResource.method_decorators</i>	
<i>ResetPasswordResource.methods</i>	A list of methods this view can handle.
<i>ResetPasswordResource.provide_automatic_options</i>	Setting this disables or force-enables the automatic options handling.
<hr/> <i>ResetPasswordResource.representations</i>	

app.blueprints.auth.ResetPasswordResource.auth_service

`ResetPasswordResource.auth_service` = <app.services.auth.AuthService object>

app.blueprints.auth.ResetPasswordResource.decorators

`ResetPasswordResource.decorators: List[Callable] = []`

The canonical way to decorate class-based views is to decorate the return value of `as_view()`. However since this moves parts of the logic from the class declaration to the place where it's hooked into the routing system.

You can place one or more decorators in this list and whenever the view function is created the result is automatically decorated.

New in version 0.8.

app.blueprints.auth.ResetPasswordResource.method_decorators

`ResetPasswordResource.method_decorators = []`

app.blueprints.auth.ResetPasswordResource.methods

`ResetPasswordResource.methods: Optional[List[str]] = {'GET', 'POST'}`
 A list of methods this view can handle.

app.blueprints.auth.ResetPasswordResource.provide_automatic_options

`ResetPasswordResource.provide_automatic_options: Optional[bool] = None`
 Setting this disables or force-enables the automatic options handling.

app.blueprints.auth.ResetPasswordResource.representations

`ResetPasswordResource.representations = None`

Methods

<i>ResetPasswordResource.__init__([api])</i>	
<i>ResetPasswordResource.as_view</i> (name, ...)	Converts the class into an actual view function that can be used with the routing system.
<i>ResetPasswordResource.dispatch_request</i> (...)	Subclasses have to override this method to implement the actual view function code.
<i>ResetPasswordResource.get</i> (token)	
<i>ResetPasswordResource.post</i> (token)	
<i>ResetPasswordResource.validate_payload</i> (func)	Perform a payload validation on expected model if necessary

app.blueprints.auth.ResetPasswordResource.__init__

`ResetPasswordResource.__init__(api=None, *args, **kwargs)`

app.blueprints.auth.ResetPasswordResource.as_view

classmethod `ResetPasswordResource.as_view(name: str, *class_args: Any, **class_kwargs: Any) → Callable`

Converts the class into an actual view function that can be used with the routing system. Internally this generates a function on the fly which will instantiate the `View` on each request and call the `dispatch_request()` method on it.

The arguments passed to `as_view()` are forwarded to the constructor of the class.

app.blueprints.auth.ResetPasswordResource.dispatch_request

`ResetPasswordResource.dispatch_request(*args, **kwargs)`

Subclasses have to override this method to implement the actual view function code. This method is called with all the arguments from the URL rule.

app.blueprints.auth.ResetPasswordResource.get

`ResetPasswordResource.get(token: str) → tuple`

app.blueprints.auth.ResetPasswordResource.post

`ResetPasswordResource.post(token: str) → tuple`

app.blueprints.auth.ResetPasswordResource.validate_payload

`ResetPasswordResource.validate_payload(func)`

Perform a payload validation on expected model if necessary

class `app.blueprints.auth.AuthBaseResource(api=None, *args, **kwargs)`

classmethod `as_view(name: str, *class_args: Any, **class_kwargs: Any) → Callable`

Converts the class into an actual view function that can be used with the routing system. Internally this generates a function on the fly which will instantiate the `View` on each request and call the `dispatch_request()` method on it.

The arguments passed to `as_view()` are forwarded to the constructor of the class.

auth_service = `<app.services.auth.AuthService object>`

decorators: `List[Callable] = []`

The canonical way to decorate class-based views is to decorate the return value of `as_view()`. However since this moves parts of the logic from the class declaration to the place where it's hooked into the routing system.

You can place one or more decorators in this list and whenever the view function is created the result is automatically decorated.

New in version 0.8.

dispatch_request(*args, **kwargs)

Subclasses have to override this method to implement the actual view function code. This method is called with all the arguments from the URL rule.


```

method_decorators = []

methods: Optional[List[str]] = None
    A list of methods this view can handle.

provide_automatic_options: Optional[bool] = None
    Setting this disables or force-enables the automatic options handling.

representations = None

validate_payload(func)
    Perform a payload validation on expected model if necessary

class app.blueprints.auth.AuthUserLoginResource(api=None, *args, **kwargs)

    classmethod as_view(name: str, *class_args: Any, **class_kwargs: Any) → Callable
        Converts the class into an actual view function that can be used with the routing system. Internally this generates a function on the fly which will instantiate the View on each request and call the dispatch_request() method on it.

        The arguments passed to as_view() are forwarded to the constructor of the class.

    auth_service = <app.services.auth.AuthService object>

    decorators: List[Callable] = []
        The canonical way to decorate class-based views is to decorate the return value of as_view(). However since this moves parts of the logic from the class declaration to the place where it's hooked into the routing system.

        You can place one or more decorators in this list and whenever the view function is created the result is automatically decorated.

        New in version 0.8.

    dispatch_request(*args, **kwargs)
        Subclasses have to override this method to implement the actual view function code. This method is called with all the arguments from the URL rule.

    method_decorators = []

    methods: Optional[List[str]] = {'POST'}
        A list of methods this view can handle.

    post() → tuple

    provide_automatic_options: Optional[bool] = None
        Setting this disables or force-enables the automatic options handling.

    representations = None

    validate_payload(func)
        Perform a payload validation on expected model if necessary

class app.blueprints.auth.AuthUserLogoutResource(api=None, *args, **kwargs)

    classmethod as_view(name: str, *class_args: Any, **class_kwargs: Any) → Callable
        Converts the class into an actual view function that can be used with the routing system. Internally this generates a function on the fly which will instantiate the View on each request and call the dispatch_request() method on it.

        The arguments passed to as_view() are forwarded to the constructor of the class.

    auth_service = <app.services.auth.AuthService object>

```

decorators: `List[Callable] = []`

The canonical way to decorate class-based views is to decorate the return value of `as_view()`. However since this moves parts of the logic from the class declaration to the place where it's hooked into the routing system.

You can place one or more decorators in this list and whenever the view function is created the result is automatically decorated.

New in version 0.8.

dispatch_request(**args, **kwargs*)

Subclasses have to override this method to implement the actual view function code. This method is called with all the arguments from the URL rule.

method_decorators = `[]`

methods: `Optional[List[str]] = {'POST'}`

A list of methods this view can handle.

post() → tuple

provide_automatic_options: `Optional[bool] = None`

Setting this disables or force-enables the automatic options handling.

representations = `None`

validate_payload(*func*)

Perform a payload validation on expected model if necessary

class `app.blueprints.auth.RequestResetPasswordResource`(*api=None, *args, **kwargs*)

classmethod **as_view**(*name: str, *class_args: Any, **class_kwargs: Any*) → `Callable`

Converts the class into an actual view function that can be used with the routing system. Internally this generates a function on the fly which will instantiate the `View` on each request and call the `dispatch_request()` method on it.

The arguments passed to `as_view()` are forwarded to the constructor of the class.

auth_service = `<app.services.auth.AuthService object>`

decorators: `List[Callable] = []`

The canonical way to decorate class-based views is to decorate the return value of `as_view()`. However since this moves parts of the logic from the class declaration to the place where it's hooked into the routing system.

You can place one or more decorators in this list and whenever the view function is created the result is automatically decorated.

New in version 0.8.

dispatch_request(**args, **kwargs*)

Subclasses have to override this method to implement the actual view function code. This method is called with all the arguments from the URL rule.

method_decorators = `[]`

methods: `Optional[List[str]] = {'POST'}`

A list of methods this view can handle.

post() → tuple

provide_automatic_options: `Optional[bool] = None`

Setting this disables or force-enables the automatic options handling.

representations = None

validate_payload(func)

Perform a payload validation on expected model if necessary

class app.blueprints.auth.**ResetPasswordResource**(*api=None, *args, **kwargs*)

classmethod as_view(*name: str, *class_args: Any, **class_kwargs: Any*) → Callable

Converts the class into an actual view function that can be used with the routing system. Internally this generates a function on the fly which will instantiate the View on each request and call the `dispatch_request()` method on it.

The arguments passed to `as_view()` are forwarded to the constructor of the class.

auth_service = <app.services.auth.AuthService object>

decorators: List[Callable] = []

The canonical way to decorate class-based views is to decorate the return value of `as_view()`. However since this moves parts of the logic from the class declaration to the place where it's hooked into the routing system.

You can place one or more decorators in this list and whenever the view function is created the result is automatically decorated.

New in version 0.8.

dispatch_request(*args, **kwargs)

Subclasses have to override this method to implement the actual view function code. This method is called with all the arguments from the URL rule.

get(*token: str*) → tuple

method_decorators = []

methods: Optional[List[str]] = {'GET', 'POST'}

A list of methods this view can handle.

post(*token: str*) → tuple

provide_automatic_options: Optional[bool] = None

Setting this disables or force-enables the automatic options handling.

representations = None

validate_payload(func)

Perform a payload validation on expected model if necessary

app.blueprints.base

Description

Classes

`BaseResource([api])`

`WelcomeResource([api])`

app.blueprints.base.BaseResource

```
class app.blueprints.base.BaseResource(api=None, *args, **kwargs)
    Bases: flask_restx.resource.Resource
```

Attributes

<i>BaseResource.decorators</i>	The canonical way to decorate class-based views is to decorate the return value of <code>as_view()</code> .
<i>BaseResource.method_decorators</i>	
<i>BaseResource.methods</i>	A list of methods this view can handle.
<i>BaseResource.provide_automatic_options</i>	Setting this disables or force-enables the automatic options handling.
<i>BaseResource.representations</i>	

app.blueprints.base.BaseResource.decorators

```
BaseResource.decorators: List[Callable] = []
```

The canonical way to decorate class-based views is to decorate the return value of `as_view()`. However since this moves parts of the logic from the class declaration to the place where it's hooked into the routing system.

You can place one or more decorators in this list and whenever the view function is created the result is automatically decorated.

New in version 0.8.

app.blueprints.base.BaseResource.method_decorators

```
BaseResource.method_decorators = []
```

app.blueprints.base.BaseResource.methods

```
BaseResource.methods: Optional[List[str]] = None
```

A list of methods this view can handle.

app.blueprints.base.BaseResource.provide_automatic_options

```
BaseResource.provide_automatic_options: Optional[bool] = None
```

Setting this disables or force-enables the automatic options handling.

app.blueprints.base.BaseResource.representations

`BaseResource.representations = None`

Methods

<code>BaseResource.__init__([api])</code>	
<code>BaseResource.as_view(name, *class_args, ...)</code>	Converts the class into an actual view function that can be used with the routing system.
<code>BaseResource.dispatch_request(*args, **kwargs)</code>	Subclasses have to override this method to implement the actual view function code.
<code>BaseResource.validate_payload(func)</code>	Perform a payload validation on expected model if necessary

app.blueprints.base.BaseResource.__init__

`BaseResource.__init__(api=None, *args, **kwargs)`

app.blueprints.base.BaseResource.as_view

classmethod `BaseResource.as_view(name: str, *class_args: Any, **class_kwargs: Any) → Callable`

Converts the class into an actual view function that can be used with the routing system. Internally this generates a function on the fly which will instantiate the `View` on each request and call the `dispatch_request()` method on it.

The arguments passed to `as_view()` are forwarded to the constructor of the class.

app.blueprints.base.BaseResource.dispatch_request

`BaseResource.dispatch_request(*args, **kwargs)`

Subclasses have to override this method to implement the actual view function code. This method is called with all the arguments from the URL rule.

app.blueprints.base.BaseResource.validate_payload

`BaseResource.validate_payload(func)`

Perform a payload validation on expected model if necessary

app.blueprints.base.WelcomeResource

```
class app.blueprints.base.WelcomeResource(api=None, *args, **kwargs)
    Bases: flask_restx.resource.Resource
```

Attributes

<i>WelcomeResource.decorators</i>	The canonical way to decorate class-based views is to decorate the return value of <code>as_view()</code> .
<i>WelcomeResource.method_decorators</i>	
<i>WelcomeResource.methods</i>	A list of methods this view can handle.
<i>WelcomeResource.provide_automatic_options</i>	Setting this disables or force-enables the automatic options handling.
<i>WelcomeResource.representations</i>	

app.blueprints.base.WelcomeResource.decorators

```
WelcomeResource.decorators: List[Callable] = []
```

The canonical way to decorate class-based views is to decorate the return value of `as_view()`. However since this moves parts of the logic from the class declaration to the place where it's hooked into the routing system.

You can place one or more decorators in this list and whenever the view function is created the result is automatically decorated.

New in version 0.8.

app.blueprints.base.WelcomeResource.method_decorators

```
WelcomeResource.method_decorators = []
```

app.blueprints.base.WelcomeResource.methods

```
WelcomeResource.methods: Optional[List[str]] = {'GET'}
```

A list of methods this view can handle.

app.blueprints.base.WelcomeResource.provide_automatic_options

```
WelcomeResource.provide_automatic_options: Optional[bool] = None
```

Setting this disables or force-enables the automatic options handling.

app.blueprints.base.WelcomeResource.representations

`WelcomeResource.representations = None`

Methods

<code>WelcomeResource.__init__([api])</code>	
<code>WelcomeResource.as_view(name, *class_args, ...)</code>	Converts the class into an actual view function that can be used with the routing system.
<code>WelcomeResource.dispatch_request(*args, **kwargs)</code>	Subclasses have to override this method to implement the actual view function code.
<code>WelcomeResource.get()</code>	
<code>WelcomeResource.validate_payload(func)</code>	Perform a payload validation on expected model if necessary

app.blueprints.base.WelcomeResource.__init__

`WelcomeResource.__init__(api=None, *args, **kwargs)`

app.blueprints.base.WelcomeResource.as_view

classmethod `WelcomeResource.as_view(name: str, *class_args: Any, **class_kwargs: Any)`
→ Callable

Converts the class into an actual view function that can be used with the routing system. Internally this generates a function on the fly which will instantiate the `View` on each request and call the `dispatch_request()` method on it.

The arguments passed to `as_view()` are forwarded to the constructor of the class.

app.blueprints.base.WelcomeResource.dispatch_request

`WelcomeResource.dispatch_request(*args, **kwargs)`

Subclasses have to override this method to implement the actual view function code. This method is called with all the arguments from the URL rule.

app.blueprints.base.WelcomeResource.get

`WelcomeResource.get()` → tuple

app.blueprints.base.WelcomeResource.validate_payload**WelcomeResource.validate_payload(func)**

Perform a payload validation on expected model if necessary

class app.blueprints.base.**BaseResource**(api=None, *args, **kwargs)**classmethod as_view**(name: str, *class_args: Any, **class_kwargs: Any) → Callable

Converts the class into an actual view function that can be used with the routing system. Internally this generates a function on the fly which will instantiate the **View** on each request and call the `dispatch_request()` method on it.

The arguments passed to `as_view()` are forwarded to the constructor of the class.

decorators: List[Callable] = []

The canonical way to decorate class-based views is to decorate the return value of `as_view()`. However since this moves parts of the logic from the class declaration to the place where it's hooked into the routing system.

You can place one or more decorators in this list and whenever the view function is created the result is automatically decorated.

New in version 0.8.

dispatch_request(*args, **kwargs)

Subclasses have to override this method to implement the actual view function code. This method is called with all the arguments from the URL rule.

method_decorators = []**methods: Optional[List[str]] = None**

A list of methods this view can handle.

provide_automatic_options: Optional[bool] = None

Setting this disables or force-enables the automatic options handling.

representations = None**validate_payload(func)**

Perform a payload validation on expected model if necessary

class app.blueprints.base.**WelcomeResource**(api=None, *args, **kwargs)**classmethod as_view**(name: str, *class_args: Any, **class_kwargs: Any) → Callable

Converts the class into an actual view function that can be used with the routing system. Internally this generates a function on the fly which will instantiate the **View** on each request and call the `dispatch_request()` method on it.

The arguments passed to `as_view()` are forwarded to the constructor of the class.

decorators: List[Callable] = []

The canonical way to decorate class-based views is to decorate the return value of `as_view()`. However since this moves parts of the logic from the class declaration to the place where it's hooked into the routing system.

You can place one or more decorators in this list and whenever the view function is created the result is automatically decorated.

New in version 0.8.

dispatch_request(*args, **kwargs)

Subclasses have to override this method to implement the actual view function code. This method is called with all the arguments from the URL rule.

get() → tuple

method_decorators = []

methods: Optional[List[str]] = {'GET'}

A list of methods this view can handle.

provide_automatic_options: Optional[bool] = None

Setting this disables or force-enables the automatic options handling.

representations = None

validate_payload(func)

Perform a payload validation on expected model if necessary

app.blueprints.documents

Description

Classes

DocumentBaseResource([api])

DocumentResource([api])

NewDocumentResource([api])

SearchDocumentResource([api])

app.blueprints.documents.DocumentBaseResource

class app.blueprints.documents.**DocumentBaseResource**(api=None, *args, **kwargs)

Bases: *app.blueprints.base.BaseResource*

Attributes

DocumentBaseResource.decorators

The canonical way to decorate class-based views is to decorate the return value of `as_view()`.

DocumentBaseResource.doc_serializer

DocumentBaseResource.doc_service

DocumentBaseResource.method_decorators

DocumentBaseResource.methods

A list of methods this view can handle.

continues on next page

Table 21 – continued from previous page

<i>DocumentBaseResource. provide_automatic_options</i>	Setting this disables or force-enables the automatic options handling.
<i>DocumentBaseResource. representations</i>	

app.blueprints.documents.DocumentBaseResource.decorators

DocumentBaseResource.decorators: `List[Callable] = []`

The canonical way to decorate class-based views is to decorate the return value of `as_view()`. However since this moves parts of the logic from the class declaration to the place where it's hooked into the routing system.

You can place one or more decorators in this list and whenever the view function is created the result is automatically decorated.

New in version 0.8.

app.blueprints.documents.DocumentBaseResource.doc_serializer

DocumentBaseResource.doc_serializer = `<DocumentSerializer(many=False)>`

app.blueprints.documents.DocumentBaseResource.doc_service

DocumentBaseResource.doc_service = `<app.services.document.DocumentService object>`

app.blueprints.documents.DocumentBaseResource.method_decorators

DocumentBaseResource.method_decorators = `[]`

app.blueprints.documents.DocumentBaseResource.methods

DocumentBaseResource.methods: `Optional[List[str]] = None`

A list of methods this view can handle.

app.blueprints.documents.DocumentBaseResource.provide_automatic_options

DocumentBaseResource.provide_automatic_options: `Optional[bool] = None`

Setting this disables or force-enables the automatic options handling.

app.blueprints.documents.DocumentBaseResource.representations

`DocumentBaseResource.representations = None`

Methods

<code>DocumentBaseResource.__init__([api])</code>	
<code>DocumentBaseResource.as_view(name, ...)</code>	Converts the class into an actual view function that can be used with the routing system.
<code>DocumentBaseResource.dispatch_request(*args, ...)</code>	Subclasses have to override this method to implement the actual view function code.
<code>DocumentBaseResource.validate_payload(func)</code>	Perform a payload validation on expected model if necessary

app.blueprints.documents.DocumentBaseResource.__init__

`DocumentBaseResource.__init__(api=None, *args, **kwargs)`

app.blueprints.documents.DocumentBaseResource.as_view

classmethod `DocumentBaseResource.as_view(name: str, *class_args: Any, **class_kwargs: Any) → Callable`

Converts the class into an actual view function that can be used with the routing system. Internally this generates a function on the fly which will instantiate the `View` on each request and call the `dispatch_request()` method on it.

The arguments passed to `as_view()` are forwarded to the constructor of the class.

app.blueprints.documents.DocumentBaseResource.dispatch_request

`DocumentBaseResource.dispatch_request(*args, **kwargs)`

Subclasses have to override this method to implement the actual view function code. This method is called with all the arguments from the URL rule.

app.blueprints.documents.DocumentBaseResource.validate_payload

`DocumentBaseResource.validate_payload(func)`

Perform a payload validation on expected model if necessary

app.blueprints.documents.DocumentResource

class app.blueprints.documents.DocumentResource(*api=None, *args, **kwargs*)

Bases: *app.blueprints.documents.DocumentBaseResource*

Attributes

<i>DocumentResource.decorators</i>	The canonical way to decorate class-based views is to decorate the return value of <i>as_view()</i> .
<i>DocumentResource.doc_serializer</i>	
<i>DocumentResource.doc_service</i>	
<i>DocumentResource.method_decorators</i>	
<i>DocumentResource.methods</i>	A list of methods this view can handle.
<i>DocumentResource.provide_automatic_options</i>	Setting this disables or force-enables the automatic options handling.
<i>DocumentResource.representations</i>	

app.blueprints.documents.DocumentResource.decorators

DocumentResource.decorators: **List[Callable] = []**

The canonical way to decorate class-based views is to decorate the return value of *as_view()*. However since this moves parts of the logic from the class declaration to the place where it's hooked into the routing system.

You can place one or more decorators in this list and whenever the view function is created the result is automatically decorated.

New in version 0.8.

app.blueprints.documents.DocumentResource.doc_serializer

DocumentResource.doc_serializer = **<DocumentSerializer(many=False)>**

app.blueprints.documents.DocumentResource.doc_service

DocumentResource.doc_service = **<app.services.document.DocumentService object>**

app.blueprints.documents.DocumentResource.method_decorators

`DocumentResource.method_decorators = []`

app.blueprints.documents.DocumentResource.methods

`DocumentResource.methods: Optional[List[str]] = {'DELETE', 'GET', 'PUT'}`
 A list of methods this view can handle.

app.blueprints.documents.DocumentResource.provide_automatic_options

`DocumentResource.provide_automatic_options: Optional[bool] = None`
 Setting this disables or force-enables the automatic options handling.

app.blueprints.documents.DocumentResource.representations

`DocumentResource.representations = None`

Methods

<hr/> <i>DocumentResource.__init__</i> ([api]) <hr/>	
<i>DocumentResource.as_view</i> (name, *class_args, ...)	Converts the class into an actual view function that can be used with the routing system.
<hr/> <i>DocumentResource.delete</i> (document_id) <hr/>	
<i>DocumentResource.dispatch_request</i> (*args, ...)	Subclasses have to override this method to implement the actual view function code.
<hr/> <i>DocumentResource.get</i> (document_id) <hr/>	
<hr/> <i>DocumentResource.put</i> (document_id) <hr/>	
<i>DocumentResource.validate_payload</i> (func)	Perform a payload validation on expected model if necessary
<hr/>	

app.blueprints.documents.DocumentResource.__init__

`DocumentResource.__init__(api=None, *args, **kwargs)`

app.blueprints.documents.DocumentResource.as_view

classmethod `DocumentResource.as_view(name: str, *class_args: Any, **class_kwargs: Any) → Callable`

Converts the class into an actual view function that can be used with the routing system. Internally this generates a function on the fly which will instantiate the `View` on each request and call the `dispatch_request()` method on it.

The arguments passed to `as_view()` are forwarded to the constructor of the class.

app.blueprints.documents.DocumentResource.delete

`DocumentResource.delete(document_id: int) → tuple`

app.blueprints.documents.DocumentResource.dispatch_request

`DocumentResource.dispatch_request(*args, **kwargs)`

Subclasses have to override this method to implement the actual view function code. This method is called with all the arguments from the URL rule.

app.blueprints.documents.DocumentResource.get

`DocumentResource.get(document_id: int) → tuple`

app.blueprints.documents.DocumentResource.put

`DocumentResource.put(document_id: int) → tuple`

app.blueprints.documents.DocumentResource.validate_payload

`DocumentResource.validate_payload(func)`

Perform a payload validation on expected model if necessary

app.blueprints.documents.NewDocumentResource

class `app.blueprints.documents.NewDocumentResource(api=None, *args, **kwargs)`

Bases: `app.blueprints.documents.DocumentBaseResource`

Attributes

<i>NewDocumentResource.decorators</i>	The canonical way to decorate class-based views is to decorate the return value of <code>as_view()</code> .
<i>NewDocumentResource.doc_serializer</i>	
<i>NewDocumentResource.doc_service</i>	
<i>NewDocumentResource.method_decorators</i>	
<i>NewDocumentResource.methods</i>	A list of methods this view can handle.
<i>NewDocumentResource.parser</i>	
<i>NewDocumentResource.provide_automatic_options</i>	Setting this disables or force-enables the automatic options handling.
<i>NewDocumentResource.representations</i>	

app.blueprints.documents.NewDocumentResource.decorators

NewDocumentResource.decorators: `List[Callable] = []`

The canonical way to decorate class-based views is to decorate the return value of `as_view()`. However since this moves parts of the logic from the class declaration to the place where it's hooked into the routing system.

You can place one or more decorators in this list and whenever the view function is created the result is automatically decorated.

New in version 0.8.

app.blueprints.documents.NewDocumentResource.doc_serializer

`NewDocumentResource.doc_serializer = <DocumentSerializer(many=False)>`

app.blueprints.documents.NewDocumentResource.doc_service

`NewDocumentResource.doc_service = <app.services.document.DocumentService object>`

app.blueprints.documents.NewDocumentResource.method_decorators

`NewDocumentResource.method_decorators = []`

app.blueprints.documents.NewDocumentResource.methods

`NewDocumentResource.methods: Optional[List[str]] = {'POST'}`
A list of methods this view can handle.

app.blueprints.documents.NewDocumentResource.parser

`NewDocumentResource.parser = <flask_restx.reqparse.RequestParser object>`

app.blueprints.documents.NewDocumentResource.provide_automatic_options

`NewDocumentResource.provide_automatic_options: Optional[bool] = None`
Setting this disables or force-enables the automatic options handling.

app.blueprints.documents.NewDocumentResource.representations

`NewDocumentResource.representations = None`

Methods

<code>NewDocumentResource.__init__([api])</code>	
<code>NewDocumentResource.as_view(name, ...)</code>	Converts the class into an actual view function that can be used with the routing system.
<code>NewDocumentResource.dispatch_request(*args, ...)</code>	Subclasses have to override this method to implement the actual view function code.
<code>NewDocumentResource.post()</code>	
<code>NewDocumentResource.validate_payload(func)</code>	Perform a payload validation on expected model if necessary

app.blueprints.documents.NewDocumentResource.__init__

`NewDocumentResource.__init__(api=None, *args, **kwargs)`

app.blueprints.documents.NewDocumentResource.as_view

classmethod `NewDocumentResource.as_view(name: str, *class_args: Any, **class_kwargs: Any) → Callable`

Converts the class into an actual view function that can be used with the routing system. Internally this generates a function on the fly which will instantiate the `View` on each request and call the `dispatch_request()` method on it.

The arguments passed to `as_view()` are forwarded to the constructor of the class.

app.blueprints.documents.NewDocumentResource.dispatch_request**NewDocumentResource.dispatch_request**(*args, **kwargs)

Subclasses have to override this method to implement the actual view function code. This method is called with all the arguments from the URL rule.

app.blueprints.documents.NewDocumentResource.post**NewDocumentResource.post**() → tuple**app.blueprints.documents.NewDocumentResource.validate_payload****NewDocumentResource.validate_payload**(func)

Perform a payload validation on expected model if necessary

app.blueprints.documents.SearchDocumentResource**class** app.blueprints.documents.**SearchDocumentResource**(api=None, *args, **kwargs)

Bases: [app.blueprints.documents.DocumentBaseResource](#)

Attributes

<i>SearchDocumentResource.decorators</i>	The canonical way to decorate class-based views is to decorate the return value of <code>as_view()</code> .
<i>SearchDocumentResource.doc_serializer</i>	
<i>SearchDocumentResource.doc_service</i>	
<i>SearchDocumentResource.method_decorators</i>	
<i>SearchDocumentResource.methods</i>	A list of methods this view can handle.
<i>SearchDocumentResource.provide_automatic_options</i>	Setting this disables or force-enables the automatic options handling.
<i>SearchDocumentResource.representations</i>	

app.blueprints.documents.SearchDocumentResource.decorators**SearchDocumentResource.decorators:** List[Callable] = []

The canonical way to decorate class-based views is to decorate the return value of `as_view()`. However since this moves parts of the logic from the class declaration to the place where it's hooked into the routing system.

You can place one or more decorators in this list and whenever the view function is created the result is automatically decorated.

New in version 0.8.

app.blueprints.documents.SearchDocumentResource.doc_serializer

```
SearchDocumentResource.doc_serializer = <DocumentSerializer(many=False)>
```

app.blueprints.documents.SearchDocumentResource.doc_service

```
SearchDocumentResource.doc_service = <app.services.document.DocumentService
object>
```

app.blueprints.documents.SearchDocumentResource.method_decorators

```
SearchDocumentResource.method_decorators = []
```

app.blueprints.documents.SearchDocumentResource.methods

```
SearchDocumentResource.methods: Optional[List[str]] = {'POST'}
```

A list of methods this view can handle.

app.blueprints.documents.SearchDocumentResource.provide_automatic_options

```
SearchDocumentResource.provide_automatic_options: Optional[bool] = None
```

Setting this disables or force-enables the automatic options handling.

app.blueprints.documents.SearchDocumentResource.representations

```
SearchDocumentResource.representations = None
```

Methods

<i>SearchDocumentResource.</i> <i>__init__</i> ([api])	
<i>SearchDocumentResource.as_view</i> (name, ...)	Converts the class into an actual view function that can be used with the routing system.
<i>SearchDocumentResource.</i> <i>dispatch_request</i> (...)	Subclasses have to override this method to im- plement the actual view function code.
<i>SearchDocumentResource.post</i> ()	
<i>SearchDocumentResource.</i> <i>validate_payload</i> (func)	Perform a payload validation on expected model if necessary

app.blueprints.documents.SearchDocumentResource.__init__

`SearchDocumentResource.__init__(api=None, *args, **kwargs)`

app.blueprints.documents.SearchDocumentResource.as_view

classmethod `SearchDocumentResource.as_view(name: str, *class_args: Any, **class_kwargs: Any) → Callable`

Converts the class into an actual view function that can be used with the routing system. Internally this generates a function on the fly which will instantiate the `View` on each request and call the `dispatch_request()` method on it.

The arguments passed to `as_view()` are forwarded to the constructor of the class.

app.blueprints.documents.SearchDocumentResource.dispatch_request

`SearchDocumentResource.dispatch_request(*args, **kwargs)`

Subclasses have to override this method to implement the actual view function code. This method is called with all the arguments from the URL rule.

app.blueprints.documents.SearchDocumentResource.post

`SearchDocumentResource.post() → tuple`

app.blueprints.documents.SearchDocumentResource.validate_payload

`SearchDocumentResource.validate_payload(func)`

Perform a payload validation on expected model if necessary

class `app.blueprints.documents.DocumentBaseResource(api=None, *args, **kwargs)`

classmethod `as_view(name: str, *class_args: Any, **class_kwargs: Any) → Callable`

Converts the class into an actual view function that can be used with the routing system. Internally this generates a function on the fly which will instantiate the `View` on each request and call the `dispatch_request()` method on it.

The arguments passed to `as_view()` are forwarded to the constructor of the class.

decorators: `List[Callable] = []`

The canonical way to decorate class-based views is to decorate the return value of `as_view()`. However since this moves parts of the logic from the class declaration to the place where it's hooked into the routing system.

You can place one or more decorators in this list and whenever the view function is created the result is automatically decorated.

New in version 0.8.

dispatch_request `(*args, **kwargs)`

Subclasses have to override this method to implement the actual view function code. This method is called with all the arguments from the URL rule.

doc_serializer = `<DocumentSerializer(many=False)>`

```
doc_service = <app.services.document.DocumentService object>
method_decorators = []
methods: Optional[List[str]] = None
    A list of methods this view can handle.
provide_automatic_options: Optional[bool] = None
    Setting this disables or force-enables the automatic options handling.
representations = None
validate_payload(func)
    Perform a payload validation on expected model if necessary
class app.blueprints.documents.DocumentResource(api=None, *args, **kwargs)

    _parser = <flask_restx.reqparse.RequestParser object>

    classmethod as_view(name: str, *class_args: Any, **class_kwargs: Any) → Callable
        Converts the class into an actual view function that can be used with the routing system. Internally this generates a function on the fly which will instantiate the View on each request and call the dispatch_request() method on it.

        The arguments passed to as_view() are forwarded to the constructor of the class.

    decorators: List[Callable] = []
        The canonical way to decorate class-based views is to decorate the return value of as_view(). However since this moves parts of the logic from the class declaration to the place where it's hooked into the routing system.

        You can place one or more decorators in this list and whenever the view function is created the result is automatically decorated.

        New in version 0.8.

    delete(document_id: int) → tuple
    dispatch_request(*args, **kwargs)
        Subclasses have to override this method to implement the actual view function code. This method is called with all the arguments from the URL rule.

    doc_serializer = <DocumentSerializer(many=False)>
    doc_service = <app.services.document.DocumentService object>
    get(document_id: int) → tuple
    method_decorators = []
    methods: Optional[List[str]] = {'DELETE', 'GET', 'PUT'}
        A list of methods this view can handle.
    provide_automatic_options: Optional[bool] = None
        Setting this disables or force-enables the automatic options handling.
    put(document_id: int) → tuple
    representations = None
    validate_payload(func)
        Perform a payload validation on expected model if necessary
class app.blueprints.documents.NewDocumentResource(api=None, *args, **kwargs)
```

classmethod as_view(*name: str, *class_args: Any, **class_kwargs: Any*) → Callable

Converts the class into an actual view function that can be used with the routing system. Internally this generates a function on the fly which will instantiate the View on each request and call the `dispatch_request()` method on it.

The arguments passed to `as_view()` are forwarded to the constructor of the class.

decorators: List[Callable] = []

The canonical way to decorate class-based views is to decorate the return value of `as_view()`. However since this moves parts of the logic from the class declaration to the place where it's hooked into the routing system.

You can place one or more decorators in this list and whenever the view function is created the result is automatically decorated.

New in version 0.8.

dispatch_request(**args, **kwargs*)

Subclasses have to override this method to implement the actual view function code. This method is called with all the arguments from the URL rule.

doc_serializer = <DocumentSerializer(many=False)>

doc_service = <app.services.document.DocumentService object>

method_decorators = []

methods: Optional[List[str]] = {'POST'}

A list of methods this view can handle.

parser = <flask_restx.reqparse.RequestParser object>

post() → tuple

provide_automatic_options: Optional[bool] = None

Setting this disables or force-enables the automatic options handling.

representations = None

validate_payload(*func*)

Perform a payload validation on expected model if necessary

class app.blueprints.documents.SearchDocumentResource(*api=None, *args, **kwargs*)

classmethod as_view(*name: str, *class_args: Any, **class_kwargs: Any*) → Callable

Converts the class into an actual view function that can be used with the routing system. Internally this generates a function on the fly which will instantiate the View on each request and call the `dispatch_request()` method on it.

The arguments passed to `as_view()` are forwarded to the constructor of the class.

decorators: List[Callable] = []

The canonical way to decorate class-based views is to decorate the return value of `as_view()`. However since this moves parts of the logic from the class declaration to the place where it's hooked into the routing system.

You can place one or more decorators in this list and whenever the view function is created the result is automatically decorated.

New in version 0.8.

dispatch_request(*args, **kwargs)

Subclasses have to override this method to implement the actual view function code. This method is called with all the arguments from the URL rule.

doc_serializer = <DocumentSerializer(many=False)>

doc_service = <app.services.document.DocumentService object>

method_decorators = []

methods: Optional[List[str]] = {'POST'}

A list of methods this view can handle.

post() → tuple

provide_automatic_options: Optional[bool] = None

Setting this disables or force-enables the automatic options handling.

representations = None

validate_payload(func)

Perform a payload validation on expected model if necessary

app.blueprints.roles

Description

Classes

NewRoleResource([api])

RoleBaseResource([api])

RoleResource([api])

RolesSearchResource([api])

app.blueprints.roles.NewRoleResource

class app.blueprints.roles.**NewRoleResource**(api=None, *args, **kwargs)

Bases: *app.blueprints.roles.RoleBaseResource*

Attributes

NewRoleResource.decorators

The canonical way to decorate class-based views is to decorate the return value of `as_view()`.

NewRoleResource.method_decorators

NewRoleResource.methods

A list of methods this view can handle.

continues on next page

Table 30 – continued from previous page

<i>NewRoleResource. provide_automatic_options</i>	Setting this disables or force-enables the automatic options handling.
<i>NewRoleResource.representations</i>	
<i>NewRoleResource.role_serializer</i>	
<i>NewRoleResource.role_service</i>	

app.blueprints.roles.NewRoleResource.decorators

NewRoleResource.decorators: `List[Callable] = []`

The canonical way to decorate class-based views is to decorate the return value of `as_view()`. However since this moves parts of the logic from the class declaration to the place where it's hooked into the routing system.

You can place one or more decorators in this list and whenever the view function is created the result is automatically decorated.

New in version 0.8.

app.blueprints.roles.NewRoleResource.method_decorators

NewRoleResource.method_decorators = `[]`

app.blueprints.roles.NewRoleResource.methods

NewRoleResource.methods: `Optional[List[str]] = {'POST'}`

A list of methods this view can handle.

app.blueprints.roles.NewRoleResource.provide_automatic_options

NewRoleResource.provide_automatic_options: `Optional[bool] = None`

Setting this disables or force-enables the automatic options handling.

app.blueprints.roles.NewRoleResource.representations

NewRoleResource.representations = `None`

app.blueprints.roles.NewRoleResource.role_serializer

```
NewRoleResource.role_serializer = <RoleSerializer(many=False)>
```

app.blueprints.roles.NewRoleResource.role_service

```
NewRoleResource.role_service = <app.services.role.RoleService object>
```

Methods

<i>NewRoleResource.__init__</i> ([api])	
<i>NewRoleResource.as_view</i> (name, *class_args, ...)	Converts the class into an actual view function that can be used with the routing system.
<i>NewRoleResource.dispatch_request</i> (*args, **kwargs)	Subclasses have to override this method to implement the actual view function code.
<i>NewRoleResource.post</i> ()	
<i>NewRoleResource.validate_payload</i> (func)	Perform a payload validation on expected model if necessary

app.blueprints.roles.NewRoleResource.__init__

```
NewRoleResource.__init__(api=None, *args, **kwargs)
```

app.blueprints.roles.NewRoleResource.as_view

```
classmethod NewRoleResource.as_view(name: str, *class_args: Any, **class_kwargs: Any)
    → Callable
```

Converts the class into an actual view function that can be used with the routing system. Internally this generates a function on the fly which will instantiate the `View` on each request and call the `dispatch_request()` method on it.

The arguments passed to `as_view()` are forwarded to the constructor of the class.

app.blueprints.roles.NewRoleResource.dispatch_request

```
NewRoleResource.dispatch_request(*args, **kwargs)
```

Subclasses have to override this method to implement the actual view function code. This method is called with all the arguments from the URL rule.

app.blueprints.roles.NewRoleResource.post

`NewRoleResource.post()` → tuple

app.blueprints.roles.NewRoleResource.validate_payload

`NewRoleResource.validate_payload(func)`
Perform a payload validation on expected model if necessary

app.blueprints.roles.RoleBaseResource

class `app.blueprints.roles.RoleBaseResource(api=None, *args, **kwargs)`
Bases: `app.blueprints.base.BaseResource`

Attributes

<code>RoleBaseResource.decorators</code>	The canonical way to decorate class-based views is to decorate the return value of <code>as_view()</code> .
<code>RoleBaseResource.method_decorators</code>	
<code>RoleBaseResource.methods</code>	A list of methods this view can handle.
<code>RoleBaseResource.provide_automatic_options</code>	Setting this disables or force-enables the automatic options handling.
<code>RoleBaseResource.representations</code>	
<code>RoleBaseResource.role_serializer</code>	
<code>RoleBaseResource.role_service</code>	

app.blueprints.roles.RoleBaseResource.decorators

`RoleBaseResource.decorators: List[Callable] = []`

The canonical way to decorate class-based views is to decorate the return value of `as_view()`. However since this moves parts of the logic from the class declaration to the place where it's hooked into the routing system.

You can place one or more decorators in this list and whenever the view function is created the result is automatically decorated.

New in version 0.8.

app.blueprints.roles.RoleBaseResource.method_decorators

```
RoleBaseResource.method_decorators = []
```

app.blueprints.roles.RoleBaseResource.methods

```
RoleBaseResource.methods: Optional[List[str]] = None
```

A list of methods this view can handle.

app.blueprints.roles.RoleBaseResource.provide_automatic_options

```
RoleBaseResource.provide_automatic_options: Optional[bool] = None
```

Setting this disables or force-enables the automatic options handling.

app.blueprints.roles.RoleBaseResource.representations

```
RoleBaseResource.representations = None
```

app.blueprints.roles.RoleBaseResource.role_serializer

```
RoleBaseResource.role_serializer = <RoleSerializer(many=False)>
```

app.blueprints.roles.RoleBaseResource.role_service

```
RoleBaseResource.role_service = <app.services.role.RoleService object>
```

Methods

<i>RoleBaseResource.__init__</i> ([api])	
<hr/>	
<i>RoleBaseResource.as_view</i> (name, *class_args, ...)	Converts the class into an actual view function that can be used with the routing system.
<i>RoleBaseResource.dispatch_request</i> (*args, ...)	Subclasses have to override this method to implement the actual view function code.
<i>RoleBaseResource.validate_payload</i> (func)	Perform a payload validation on expected model if necessary

app.blueprints.roles.RoleBaseResource.__init__

`RoleBaseResource.__init__(api=None, *args, **kwargs)`

app.blueprints.roles.RoleBaseResource.as_view

classmethod `RoleBaseResource.as_view(name: str, *class_args: Any, **class_kwargs: Any) → Callable`

Converts the class into an actual view function that can be used with the routing system. Internally this generates a function on the fly which will instantiate the `View` on each request and call the `dispatch_request()` method on it.

The arguments passed to `as_view()` are forwarded to the constructor of the class.

app.blueprints.roles.RoleBaseResource.dispatch_request

`RoleBaseResource.dispatch_request(*args, **kwargs)`

Subclasses have to override this method to implement the actual view function code. This method is called with all the arguments from the URL rule.

app.blueprints.roles.RoleBaseResource.validate_payload

`RoleBaseResource.validate_payload(func)`

Perform a payload validation on expected model if necessary

app.blueprints.roles.RoleResource

class `app.blueprints.roles.RoleResource(api=None, *args, **kwargs)`

Bases: `app.blueprints.roles.RoleBaseResource`

Attributes

<code>RoleResource.decorators</code>	The canonical way to decorate class-based views is to decorate the return value of <code>as_view()</code> .
<code>RoleResource.method_decorators</code>	
<code>RoleResource.methods</code>	A list of methods this view can handle.
<code>RoleResource.provide_automatic_options</code>	Setting this disables or force-enables the automatic options handling.
<code>RoleResource.representations</code>	
<code>RoleResource.role_serializer</code>	
<code>RoleResource.role_service</code>	

app.blueprints.roles.RoleResource.decorators

RoleResource.decorators: `List[Callable] = []`

The canonical way to decorate class-based views is to decorate the return value of `as_view()`. However since this moves parts of the logic from the class declaration to the place where it's hooked into the routing system.

You can place one or more decorators in this list and whenever the view function is created the result is automatically decorated.

New in version 0.8.

app.blueprints.roles.RoleResource.method_decorators

RoleResource.method_decorators = `[]`

app.blueprints.roles.RoleResource.methods

RoleResource.methods: `Optional[List[str]] = {'DELETE', 'GET', 'PUT'}`

A list of methods this view can handle.

app.blueprints.roles.RoleResource.provide_automatic_options

RoleResource.provide_automatic_options: `Optional[bool] = None`

Setting this disables or force-enables the automatic options handling.

app.blueprints.roles.RoleResource.representations

RoleResource.representations = `None`

app.blueprints.roles.RoleResource.role_serializer

RoleResource.role_serializer = `<RoleSerializer(many=False)>`

app.blueprints.roles.RoleResource.role_service

RoleResource.role_service = `<app.services.role.RoleService object>`

Methods

RoleResource.__init__([api])

RoleResource.as_view(name, *class_args, ...) Converts the class into an actual view function that can be used with the routing system.

RoleResource.delete(role_id)

continues on next page

Table 35 – continued from previous page

<code>RoleResource.dispatch_request(*args, **kwargs)</code>	Subclasses have to override this method to implement the actual view function code.
<code>RoleResource.get(role_id)</code>	
<code>RoleResource.put(role_id)</code>	
<code>RoleResource.validate_payload(func)</code>	Perform a payload validation on expected model if necessary

app.blueprints.roles.RoleResource.__init__

`RoleResource.__init__(api=None, *args, **kwargs)`

app.blueprints.roles.RoleResource.as_view

classmethod `RoleResource.as_view(name: str, *class_args: Any, **class_kwargs: Any) → Callable`

Converts the class into an actual view function that can be used with the routing system. Internally this generates a function on the fly which will instantiate the `View` on each request and call the `dispatch_request()` method on it.

The arguments passed to `as_view()` are forwarded to the constructor of the class.

app.blueprints.roles.RoleResource.delete

`RoleResource.delete(role_id: int) → tuple`

app.blueprints.roles.RoleResource.dispatch_request

`RoleResource.dispatch_request(*args, **kwargs)`

Subclasses have to override this method to implement the actual view function code. This method is called with all the arguments from the URL rule.

app.blueprints.roles.RoleResource.get

`RoleResource.get(role_id: int) → tuple`

app.blueprints.roles.RoleResource.put

`RoleResource.put(role_id: int) → tuple`

app.blueprints.roles.RoleResource.validate_payload

`RoleResource.validate_payload(func)`
Perform a payload validation on expected model if necessary

app.blueprints.roles.RolesSearchResource

class `app.blueprints.roles.RolesSearchResource(api=None, *args, **kwargs)`
Bases: `app.blueprints.roles.RoleBaseResource`

Attributes

<code>RolesSearchResource.decorators</code>	The canonical way to decorate class-based views is to decorate the return value of <code>as_view()</code> .
<code>RolesSearchResource.method_decorators</code>	
<code>RolesSearchResource.methods</code>	A list of methods this view can handle.
<code>RolesSearchResource.provide_automatic_options</code>	Setting this disables or force-enables the automatic options handling.
<code>RolesSearchResource.representations</code>	
<code>RolesSearchResource.role_serializer</code>	
<code>RolesSearchResource.role_service</code>	

app.blueprints.roles.RolesSearchResource.decorators

`RolesSearchResource.decorators: List[Callable] = []`

The canonical way to decorate class-based views is to decorate the return value of `as_view()`. However since this moves parts of the logic from the class declaration to the place where it's hooked into the routing system.

You can place one or more decorators in this list and whenever the view function is created the result is automatically decorated.

New in version 0.8.

app.blueprints.roles.RolesSearchResource.method_decorators

`RolesSearchResource.method_decorators = []`

app.blueprints.roles.RolesSearchResource.methods

`RolesSearchResource.methods: Optional[List[str]] = {'POST'}`
 A list of methods this view can handle.

app.blueprints.roles.RolesSearchResource.provide_automatic_options

`RolesSearchResource.provide_automatic_options: Optional[bool] = None`
 Setting this disables or force-enables the automatic options handling.

app.blueprints.roles.RolesSearchResource.representations

`RolesSearchResource.representations = None`

app.blueprints.roles.RolesSearchResource.role_serializer

`RolesSearchResource.role_serializer = <RoleSerializer(many=False)>`

app.blueprints.roles.RolesSearchResource.role_service

`RolesSearchResource.role_service = <app.services.role.RoleService object>`

Methods

<code>RolesSearchResource.__init__([api])</code>	
<code>RolesSearchResource.as_view(name, ...)</code>	Converts the class into an actual view function that can be used with the routing system.
<code>RolesSearchResource.dispatch_request(*args, ...)</code>	Subclasses have to override this method to implement the actual view function code.
<code>RolesSearchResource.post()</code>	
<code>RolesSearchResource.validate_payload(func)</code>	Perform a payload validation on expected model if necessary

app.blueprints.roles.RolesSearchResource.__init__

`RolesSearchResource.__init__(api=None, *args, **kwargs)`

app.blueprints.roles.RolesSearchResource.as_view

classmethod `RolesSearchResource.as_view(name: str, *class_args: Any, **class_kwargs: Any) → Callable`

Converts the class into an actual view function that can be used with the routing system. Internally this generates a function on the fly which will instantiate the `View` on each request and call the `dispatch_request()` method on it.

The arguments passed to `as_view()` are forwarded to the constructor of the class.

app.blueprints.roles.RolesSearchResource.dispatch_request

`RolesSearchResource.dispatch_request(*args, **kwargs)`

Subclasses have to override this method to implement the actual view function code. This method is called with all the arguments from the URL rule.

app.blueprints.roles.RolesSearchResource.post

`RolesSearchResource.post() → tuple`

app.blueprints.roles.RolesSearchResource.validate_payload

`RolesSearchResource.validate_payload(func)`

Perform a payload validation on expected model if necessary

class `app.blueprints.roles.NewRoleResource(api=None, *args, **kwargs)`

classmethod `as_view(name: str, *class_args: Any, **class_kwargs: Any) → Callable`

Converts the class into an actual view function that can be used with the routing system. Internally this generates a function on the fly which will instantiate the `View` on each request and call the `dispatch_request()` method on it.

The arguments passed to `as_view()` are forwarded to the constructor of the class.

decorators: `List[Callable] = []`

The canonical way to decorate class-based views is to decorate the return value of `as_view()`. However since this moves parts of the logic from the class declaration to the place where it's hooked into the routing system.

You can place one or more decorators in this list and whenever the view function is created the result is automatically decorated.

New in version 0.8.

dispatch_request `(*args, **kwargs)`

Subclasses have to override this method to implement the actual view function code. This method is called with all the arguments from the URL rule.

method_decorators `= []`

methods: `Optional[List[str]] = {'POST'}`

A list of methods this view can handle.

post() `→ tuple`

provide_automatic_options: `Optional[bool] = None`

Setting this disables or force-enables the automatic options handling.

representations = `None`

role_serializer = `<RoleSerializer(many=False)>`

role_service = `<app.services.role.RoleService object>`

validate_payload(*func*)

Perform a payload validation on expected model if necessary

class `app.blueprints.roles.RoleBaseResource`(*api=None, *args, **kwargs*)

classmethod `as_view`(*name: str, *class_args: Any, **class_kwargs: Any*) `→ Callable`

Converts the class into an actual view function that can be used with the routing system. Internally this generates a function on the fly which will instantiate the `View` on each request and call the `dispatch_request()` method on it.

The arguments passed to `as_view()` are forwarded to the constructor of the class.

decorators: `List[Callable] = []`

The canonical way to decorate class-based views is to decorate the return value of `as_view()`. However since this moves parts of the logic from the class declaration to the place where it's hooked into the routing system.

You can place one or more decorators in this list and whenever the view function is created the result is automatically decorated.

New in version 0.8.

dispatch_request(**args, **kwargs*)

Subclasses have to override this method to implement the actual view function code. This method is called with all the arguments from the URL rule.

method_decorators = `[]`

methods: `Optional[List[str]] = None`

A list of methods this view can handle.

provide_automatic_options: `Optional[bool] = None`

Setting this disables or force-enables the automatic options handling.

representations = `None`

role_serializer = `<RoleSerializer(many=False)>`

role_service = `<app.services.role.RoleService object>`

validate_payload(*func*)

Perform a payload validation on expected model if necessary

class `app.blueprints.roles.RoleResource`(*api=None, *args, **kwargs*)

classmethod `as_view`(*name: str, *class_args: Any, **class_kwargs: Any*) `→ Callable`

Converts the class into an actual view function that can be used with the routing system. Internally this generates a function on the fly which will instantiate the `View` on each request and call the `dispatch_request()` method on it.

The arguments passed to `as_view()` are forwarded to the constructor of the class.

decorators: `List[Callable] = []`

The canonical way to decorate class-based views is to decorate the return value of `as_view()`. However since this moves parts of the logic from the class declaration to the place where it's hooked into the routing system.

You can place one or more decorators in this list and whenever the view function is created the result is automatically decorated.

New in version 0.8.

delete(*role_id: int*) → tuple

dispatch_request(**args, **kwargs*)

Subclasses have to override this method to implement the actual view function code. This method is called with all the arguments from the URL rule.

get(*role_id: int*) → tuple

method_decorators = []

methods: `Optional[List[str]] = {'DELETE', 'GET', 'PUT'}`

A list of methods this view can handle.

provide_automatic_options: `Optional[bool] = None`

Setting this disables or force-enables the automatic options handling.

put(*role_id: int*) → tuple

representations = None

role_serializer = `<RoleSerializer(many=False)>`

role_service = `<app.services.role.RoleService object>`

validate_payload(*func*)

Perform a payload validation on expected model if necessary

class `app.blueprints.roles.RolesSearchResource`(*api=None, *args, **kwargs*)

classmethod **as_view**(*name: str, *class_args: Any, **class_kwargs: Any*) → Callable

Converts the class into an actual view function that can be used with the routing system. Internally this generates a function on the fly which will instantiate the View on each request and call the `dispatch_request()` method on it.

The arguments passed to `as_view()` are forwarded to the constructor of the class.

decorators: `List[Callable] = []`

The canonical way to decorate class-based views is to decorate the return value of `as_view()`. However since this moves parts of the logic from the class declaration to the place where it's hooked into the routing system.

You can place one or more decorators in this list and whenever the view function is created the result is automatically decorated.

New in version 0.8.

dispatch_request(**args, **kwargs*)

Subclasses have to override this method to implement the actual view function code. This method is called with all the arguments from the URL rule.

method_decorators = []

methods: `Optional[List[str]] = {'POST'}`
 A list of methods this view can handle.

post() `→ tuple`

provide_automatic_options: `Optional[bool] = None`
 Setting this disables or force-enables the automatic options handling.

representations `= None`

role_serializer `= <RoleSerializer(many=False)>`

role_service `= <app.services.role.RoleService object>`

validate_payload `(func)`
 Perform a payload validation on expected model if necessary

app.blueprints.tasks

Description

Classes

TaskResource([api])

TaskStatusResource([api])

app.blueprints.tasks.TaskResource

class `app.blueprints.tasks.TaskResource`(*api=None, *args, **kwargs*)
 Bases: `flask_restx.resource.Resource`

Attributes

<i>TaskResource.decorators</i>	The canonical way to decorate class-based views is to decorate the return value of <code>as_view()</code> .
<i>TaskResource.method_decorators</i>	
<i>TaskResource.methods</i>	A list of methods this view can handle.
<i>TaskResource.provide_automatic_options</i>	Setting this disables or force-enables the automatic options handling.
<i>TaskResource.representations</i>	
<i>TaskResource.task_service</i>	

app.blueprints.tasks.TaskResource.decorators

TaskResource.decorators: `List[Callable] = []`

The canonical way to decorate class-based views is to decorate the return value of `as_view()`. However since this moves parts of the logic from the class declaration to the place where it's hooked into the routing system.

You can place one or more decorators in this list and whenever the view function is created the result is automatically decorated.

New in version 0.8.

app.blueprints.tasks.TaskResource.method_decorators

TaskResource.method_decorators = `[]`

app.blueprints.tasks.TaskResource.methods

TaskResource.methods: `Optional[List[str]] = None`

A list of methods this view can handle.

app.blueprints.tasks.TaskResource.provide_automatic_options

TaskResource.provide_automatic_options: `Optional[bool] = None`

Setting this disables or force-enables the automatic options handling.

app.blueprints.tasks.TaskResource.representations

TaskResource.representations = `None`

app.blueprints.tasks.TaskResource.task_service

TaskResource.task_service = `<app.services.task.TaskService object>`

Methods

<code>TaskResource.__init__([api])</code>	
<hr/>	
<code>TaskResource.as_view(name, *class_args, ...)</code>	Converts the class into an actual view function that can be used with the routing system.
<code>TaskResource.dispatch_request(*args, **kwargs)</code>	Subclasses have to override this method to implement the actual view function code.
<code>TaskResource.validate_payload(func)</code>	Perform a payload validation on expected model if necessary

app.blueprints.tasks.TaskResource.__init__

`TaskResource.__init__(api=None, *args, **kwargs)`

app.blueprints.tasks.TaskResource.as_view

classmethod `TaskResource.as_view(name: str, *class_args: Any, **class_kwargs: Any) → Callable`

Converts the class into an actual view function that can be used with the routing system. Internally this generates a function on the fly which will instantiate the `View` on each request and call the `dispatch_request()` method on it.

The arguments passed to `as_view()` are forwarded to the constructor of the class.

app.blueprints.tasks.TaskResource.dispatch_request

`TaskResource.dispatch_request(*args, **kwargs)`

Subclasses have to override this method to implement the actual view function code. This method is called with all the arguments from the URL rule.

app.blueprints.tasks.TaskResource.validate_payload

`TaskResource.validate_payload(func)`

Perform a payload validation on expected model if necessary

app.blueprints.tasks.TaskStatusResource

class `app.blueprints.tasks.TaskStatusResource(api=None, *args, **kwargs)`

Bases: `app.blueprints.tasks.TaskResource`

Attributes

<code>TaskStatusResource.decorators</code>	The canonical way to decorate class-based views is to decorate the return value of <code>as_view()</code> .
<code>TaskStatusResource.method_decorators</code>	
<code>TaskStatusResource.methods</code>	A list of methods this view can handle.
<code>TaskStatusResource.provide_automatic_options</code>	Setting this disables or force-enables the automatic options handling.
<code>TaskStatusResource.representations</code>	
<code>TaskStatusResource.task_service</code>	

app.blueprints.tasks.TaskStatusResource.decorators

TaskStatusResource.decorators: `List[Callable] = []`

The canonical way to decorate class-based views is to decorate the return value of `as_view()`. However since this moves parts of the logic from the class declaration to the place where it's hooked into the routing system.

You can place one or more decorators in this list and whenever the view function is created the result is automatically decorated.

New in version 0.8.

app.blueprints.tasks.TaskStatusResource.method_decorators

TaskStatusResource.method_decorators = `[]`

app.blueprints.tasks.TaskStatusResource.methods

TaskStatusResource.methods: `Optional[List[str]] = {'GET'}`

A list of methods this view can handle.

app.blueprints.tasks.TaskStatusResource.provide_automatic_options

TaskStatusResource.provide_automatic_options: `Optional[bool] = None`

Setting this disables or force-enables the automatic options handling.

app.blueprints.tasks.TaskStatusResource.representations

TaskStatusResource.representations = `None`

app.blueprints.tasks.TaskStatusResource.task_service

TaskStatusResource.task_service = `<app.services.task.TaskService object>`

Methods

<code>TaskStatusResource.__init__([api])</code>	
<code>TaskStatusResource.as_view(name, ...)</code>	Converts the class into an actual view function that can be used with the routing system.
<code>TaskStatusResource.dispatch_request(*args, ...)</code>	Subclasses have to override this method to implement the actual view function code.
<code>TaskStatusResource.get(task_id)</code>	
<code>TaskStatusResource.validate_payload(func)</code>	Perform a payload validation on expected model if necessary

app.blueprints.tasks.TaskStatusResource.__init__

`TaskStatusResource.__init__(api=None, *args, **kwargs)`

app.blueprints.tasks.TaskStatusResource.as_view

classmethod `TaskStatusResource.as_view(name: str, *class_args: Any, **class_kwargs: Any) → Callable`

Converts the class into an actual view function that can be used with the routing system. Internally this generates a function on the fly which will instantiate the `View` on each request and call the `dispatch_request()` method on it.

The arguments passed to `as_view()` are forwarded to the constructor of the class.

app.blueprints.tasks.TaskStatusResource.dispatch_request

`TaskStatusResource.dispatch_request(*args, **kwargs)`

Subclasses have to override this method to implement the actual view function code. This method is called with all the arguments from the URL rule.

app.blueprints.tasks.TaskStatusResource.get

`TaskStatusResource.get(task_id: str)`

app.blueprints.tasks.TaskStatusResource.validate_payload

`TaskStatusResource.validate_payload(func)`

Perform a payload validation on expected model if necessary

class `app.blueprints.tasks.TaskResource(api=None, *args, **kwargs)`

classmethod `as_view(name: str, *class_args: Any, **class_kwargs: Any) → Callable`

Converts the class into an actual view function that can be used with the routing system. Internally this generates a function on the fly which will instantiate the `View` on each request and call the `dispatch_request()` method on it.

The arguments passed to `as_view()` are forwarded to the constructor of the class.

decorators: `List[Callable] = []`

The canonical way to decorate class-based views is to decorate the return value of `as_view()`. However since this moves parts of the logic from the class declaration to the place where it's hooked into the routing system.

You can place one or more decorators in this list and whenever the view function is created the result is automatically decorated.

New in version 0.8.

dispatch_request `(*args, **kwargs)`

Subclasses have to override this method to implement the actual view function code. This method is called with all the arguments from the URL rule.

method_decorators `= []`

methods: `Optional[List[str]] = None`

A list of methods this view can handle.

provide_automatic_options: `Optional[bool] = None`

Setting this disables or force-enables the automatic options handling.

representations = `None`

task_service = `<app.services.task.TaskService object>`

validate_payload(*func*)

Perform a payload validation on expected model if necessary

class `app.blueprints.tasks.TaskStatusResource`(*api=None, *args, **kwargs*)

classmethod `as_view`(*name: str, *class_args: Any, **class_kwargs: Any*) → `Callable`

Converts the class into an actual view function that can be used with the routing system. Internally this generates a function on the fly which will instantiate the View on each request and call the `dispatch_request()` method on it.

The arguments passed to `as_view()` are forwarded to the constructor of the class.

decorators: `List[Callable] = []`

The canonical way to decorate class-based views is to decorate the return value of `as_view()`. However since this moves parts of the logic from the class declaration to the place where it's hooked into the routing system.

You can place one or more decorators in this list and whenever the view function is created the result is automatically decorated.

New in version 0.8.

dispatch_request(**args, **kwargs*)

Subclasses have to override this method to implement the actual view function code. This method is called with all the arguments from the URL rule.

get(*task_id: str*)

method_decorators = `[]`

methods: `Optional[List[str]] = {'GET'}`

A list of methods this view can handle.

provide_automatic_options: `Optional[bool] = None`

Setting this disables or force-enables the automatic options handling.

representations = `None`

task_service = `<app.services.task.TaskService object>`

validate_payload(*func*)

Perform a payload validation on expected model if necessary

app.blueprints.users**Description****Classes**

ExportUsersExcelAndWordResource([api])

ExportUsersExcelResource([api])

ExportUsersWordResource([api])

NewUserResource([api])

UserBaseResource([api])

UserResource([api])

UsersSearchResource([api])

app.blueprints.users.ExportUsersExcelAndWordResource**class** app.blueprints.users.**ExportUsersExcelAndWordResource**(api=None, *args, **kwargs)Bases: *app.blueprints.users.UserBaseResource***Attributes**

<i>ExportUsersExcelAndWordResource.decorators</i>	The canonical way to decorate class-based views is to decorate the return value of <code>as_view()</code> .
<i>ExportUsersExcelAndWordResource.method_decorators</i>	
<i>ExportUsersExcelAndWordResource.methods</i>	A list of methods this view can handle.
<i>ExportUsersExcelAndWordResource.provide_automatic_options</i>	Setting this disables or force-enables the automatic options handling.
<i>ExportUsersExcelAndWordResource.representations</i>	
<i>ExportUsersExcelAndWordResource.task_service</i>	
<i>ExportUsersExcelAndWordResource.user_serializer</i>	
<i>ExportUsersExcelAndWordResource.user_service</i>	

app.blueprints.users.ExportUsersExcelAndWordResource.decorators

```
ExportUsersExcelAndWordResource.decorators: List[Callable] = []
```

The canonical way to decorate class-based views is to decorate the return value of `as_view()`. However since this moves parts of the logic from the class declaration to the place where it's hooked into the routing system.

You can place one or more decorators in this list and whenever the view function is created the result is automatically decorated.

New in version 0.8.

app.blueprints.users.ExportUsersExcelAndWordResource.method_decorators

```
ExportUsersExcelAndWordResource.method_decorators = []
```

app.blueprints.users.ExportUsersExcelAndWordResource.methods

```
ExportUsersExcelAndWordResource.methods: Optional[List[str]] = {'POST'}
```

A list of methods this view can handle.

app.blueprints.users.ExportUsersExcelAndWordResource.provide_automatic_options

```
ExportUsersExcelAndWordResource.provide_automatic_options: Optional[bool] = None
```

Setting this disables or force-enables the automatic options handling.

app.blueprints.users.ExportUsersExcelAndWordResource.representations

```
ExportUsersExcelAndWordResource.representations = None
```

app.blueprints.users.ExportUsersExcelAndWordResource.task_service

```
ExportUsersExcelAndWordResource.task_service =  
<app.services.task.TaskService object>
```

app.blueprints.users.ExportUsersExcelAndWordResource.user_serializer

```
ExportUsersExcelAndWordResource.user_serializer =  
<UserSerializer(many=False)>
```

app.blueprints.users.ExportUsersExcelAndWordResource.user_service

```
ExportUsersExcelAndWordResource.user_service =
<app.services.user.UserService object>
```

Methods

<i>ExportUsersExcelAndWordResource.__init__([api])</i>	
<i>ExportUsersExcelAndWordResource.as_view(...)</i>	Converts the class into an actual view function that can be used with the routing system.
<i>ExportUsersExcelAndWordResource.dispatch_request(...)</i>	Subclasses have to override this method to implement the actual view function code.
<i>ExportUsersExcelAndWordResource.post()</i>	
<i>ExportUsersExcelAndWordResource.validate_payload(func)</i>	Perform a payload validation on expected model if necessary

app.blueprints.users.ExportUsersExcelAndWordResource.__init__

```
ExportUsersExcelAndWordResource.__init__(api=None, *args, **kwargs)
```

app.blueprints.users.ExportUsersExcelAndWordResource.as_view

```
classmethod ExportUsersExcelAndWordResource.as_view(name: str, *class_args: Any,
                                                    **class_kwargs: Any) →
                                                    Callable
```

Converts the class into an actual view function that can be used with the routing system. Internally this generates a function on the fly which will instantiate the View on each request and call the *dispatch_request()* method on it.

The arguments passed to *as_view()* are forwarded to the constructor of the class.

app.blueprints.users.ExportUsersExcelAndWordResource.dispatch_request

`ExportUsersExcelAndWordResource.dispatch_request(*args, **kwargs)`

Subclasses have to override this method to implement the actual view function code. This method is called with all the arguments from the URL rule.

app.blueprints.users.ExportUsersExcelAndWordResource.post

`ExportUsersExcelAndWordResource.post()` → tuple

app.blueprints.users.ExportUsersExcelAndWordResource.validate_payload

`ExportUsersExcelAndWordResource.validate_payload(func)`

Perform a payload validation on expected model if necessary

app.blueprints.users.ExportUsersExcelResource

class `app.blueprints.users.ExportUsersExcelResource(api=None, *args, **kwargs)`

Bases: `app.blueprints.users.UserBaseResource`

Attributes

<code>ExportUsersExcelResource.decorators</code>	The canonical way to decorate class-based views is to decorate the return value of <code>as_view()</code> .
<code>ExportUsersExcelResource.method_decorators</code>	
<code>ExportUsersExcelResource.methods</code>	A list of methods this view can handle.
<code>ExportUsersExcelResource.provide_automatic_options</code>	Setting this disables or force-enables the automatic options handling.
<code>ExportUsersExcelResource.representations</code>	
<code>ExportUsersExcelResource.task_service</code>	
<code>ExportUsersExcelResource.user_serializer</code>	
<code>ExportUsersExcelResource.user_service</code>	

app.blueprints.users.ExportUsersExcelResource.decorators

`ExportUsersExcelResource.decorators: List[Callable] = []`

The canonical way to decorate class-based views is to decorate the return value of `as_view()`. However since this moves parts of the logic from the class declaration to the place where it's hooked into the routing system.

You can place one or more decorators in this list and whenever the view function is created the result is automatically decorated.

New in version 0.8.

app.blueprints.users.ExportUsersExcelResource.method_decorators

`ExportUsersExcelResource.method_decorators = []`

app.blueprints.users.ExportUsersExcelResource.methods

`ExportUsersExcelResource.methods: Optional[List[str]] = {'POST'}`

A list of methods this view can handle.

app.blueprints.users.ExportUsersExcelResource.provide_automatic_options

`ExportUsersExcelResource.provide_automatic_options: Optional[bool] = None`

Setting this disables or force-enables the automatic options handling.

app.blueprints.users.ExportUsersExcelResource.representations

`ExportUsersExcelResource.representations = None`

app.blueprints.users.ExportUsersExcelResource.task_service

`ExportUsersExcelResource.task_service = <app.services.task.TaskService object>`

app.blueprints.users.ExportUsersExcelResource.user_serializer

`ExportUsersExcelResource.user_serializer = <UserSerializer(many=False)>`

app.blueprints.users.ExportUsersExcelResource.user_service

`ExportUsersExcelResource.user_service = <app.services.user.UserService object>`

Methods

<code>ExportUsersExcelResource.__init__([api])</code>	
<code>ExportUsersExcelResource.as_view(name, ...)</code>	Converts the class into an actual view function that can be used with the routing system.
<code>ExportUsersExcelResource.dispatch_request(...)</code>	Subclasses have to override this method to implement the actual view function code.
<code>ExportUsersExcelResource.post()</code>	
<code>ExportUsersExcelResource.validate_payload(func)</code>	Perform a payload validation on expected model if necessary

app.blueprints.users.ExportUsersExcelResource.__init__

`ExportUsersExcelResource.__init__(api=None, *args, **kwargs)`

app.blueprints.users.ExportUsersExcelResource.as_view

classmethod `ExportUsersExcelResource.as_view(name: str, *class_args: Any, **class_kwargs: Any) → Callable`

Converts the class into an actual view function that can be used with the routing system. Internally this generates a function on the fly which will instantiate the `View` on each request and call the `dispatch_request()` method on it.

The arguments passed to `as_view()` are forwarded to the constructor of the class.

app.blueprints.users.ExportUsersExcelResource.dispatch_request

`ExportUsersExcelResource.dispatch_request(*args, **kwargs)`

Subclasses have to override this method to implement the actual view function code. This method is called with all the arguments from the URL rule.

app.blueprints.users.ExportUsersExcelResource.post

`ExportUsersExcelResource.post() → tuple`

app.blueprints.users.ExportUsersExcelResource.validate_payload

`ExportUsersExcelResource.validate_payload(func)`
 Perform a payload validation on expected model if necessary

app.blueprints.users.ExportUsersWordResource

class `app.blueprints.users.ExportUsersWordResource`(*api=None, *args, **kwargs*)
 Bases: `app.blueprints.users.UserBaseResource`

Attributes

<code>ExportUsersWordResource.decorators</code>	The canonical way to decorate class-based views is to decorate the return value of <code>as_view()</code> .
<code>ExportUsersWordResource.method_decorators</code>	
<code>ExportUsersWordResource.methods</code>	A list of methods this view can handle.
<code>ExportUsersWordResource.provide_automatic_options</code>	Setting this disables or force-enables the automatic options handling.
<code>ExportUsersWordResource.representations</code>	
<code>ExportUsersWordResource.task_service</code>	
<code>ExportUsersWordResource.user_serializer</code>	
<code>ExportUsersWordResource.user_service</code>	

app.blueprints.users.ExportUsersWordResource.decorators

`ExportUsersWordResource.decorators: List[Callable] = []`

The canonical way to decorate class-based views is to decorate the return value of `as_view()`. However since this moves parts of the logic from the class declaration to the place where it's hooked into the routing system.

You can place one or more decorators in this list and whenever the view function is created the result is automatically decorated.

New in version 0.8.

app.blueprints.users.ExportUsersWordResource.method_decorators

```
ExportUsersWordResource.method_decorators = []
```

app.blueprints.users.ExportUsersWordResource.methods

```
ExportUsersWordResource.methods: Optional[List[str]] = {'POST'}
```

A list of methods this view can handle.

app.blueprints.users.ExportUsersWordResource.provide_automatic_options

```
ExportUsersWordResource.provide_automatic_options: Optional[bool] = None
```

Setting this disables or force-enables the automatic options handling.

app.blueprints.users.ExportUsersWordResource.representations

```
ExportUsersWordResource.representations = None
```

app.blueprints.users.ExportUsersWordResource.task_service

```
ExportUsersWordResource.task_service = <app.services.task.TaskService
object>
```

app.blueprints.users.ExportUsersWordResource.user_serializer

```
ExportUsersWordResource.user_serializer = <UserSerializer(many=False)>
```

app.blueprints.users.ExportUsersWordResource.user_service

```
ExportUsersWordResource.user_service = <app.services.user.UserService
object>
```

Methods

<i>ExportUsersWordResource.</i> <i>__init__([api])</i>	
<i>ExportUsersWordResource.</i> <i>as_view(name, ...)</i>	Converts the class into an actual view function that can be used with the routing system.
<i>ExportUsersWordResource.</i> <i>dispatch_request(...)</i>	Subclasses have to override this method to implement the actual view function code.
<i>ExportUsersWordResource.post()</i>	
<i>ExportUsersWordResource.</i> <i>validate_payload(func)</i>	Perform a payload validation on expected model if necessary

app.blueprints.users.ExportUsersWordResource.__init__

`ExportUsersWordResource.__init__(api=None, *args, **kwargs)`

app.blueprints.users.ExportUsersWordResource.as_view

classmethod `ExportUsersWordResource.as_view(name: str, *class_args: Any, **class_kwargs: Any) → Callable`

Converts the class into an actual view function that can be used with the routing system. Internally this generates a function on the fly which will instantiate the `View` on each request and call the `dispatch_request()` method on it.

The arguments passed to `as_view()` are forwarded to the constructor of the class.

app.blueprints.users.ExportUsersWordResource.dispatch_request

`ExportUsersWordResource.dispatch_request(*args, **kwargs)`

Subclasses have to override this method to implement the actual view function code. This method is called with all the arguments from the URL rule.

app.blueprints.users.ExportUsersWordResource.post

`ExportUsersWordResource.post() → tuple`

app.blueprints.users.ExportUsersWordResource.validate_payload

`ExportUsersWordResource.validate_payload(func)`

Perform a payload validation on expected model if necessary

app.blueprints.users.NewUserResource

class `app.blueprints.users.NewUserResource(api=None, *args, **kwargs)`

Bases: `app.blueprints.users.UserBaseResource`

Attributes

<code>NewUserResource.decorators</code>	The canonical way to decorate class-based views is to decorate the return value of <code>as_view()</code> .
<code>NewUserResource.method_decorators</code>	
<code>NewUserResource.methods</code>	A list of methods this view can handle.
<code>NewUserResource.provide_automatic_options</code>	Setting this disables or force-enables the automatic options handling.
<code>NewUserResource.representations</code>	

continues on next page

Table 50 – continued from previous page

NewUserResource.task_service

NewUserResource.user_serializer

NewUserResource.user_service

app.blueprints.users.NewUserResource.decorators**NewUserResource.decorators:** `List[Callable] = []`

The canonical way to decorate class-based views is to decorate the return value of `as_view()`. However since this moves parts of the logic from the class declaration to the place where it's hooked into the routing system.

You can place one or more decorators in this list and whenever the view function is created the result is automatically decorated.

New in version 0.8.

app.blueprints.users.NewUserResource.method_decorators**NewUserResource.method_decorators** = `[]`**app.blueprints.users.NewUserResource.methods****NewUserResource.methods:** `Optional[List[str]] = {'POST'}`

A list of methods this view can handle.

app.blueprints.users.NewUserResource.provide_automatic_options**NewUserResource.provide_automatic_options:** `Optional[bool] = None`

Setting this disables or force-enables the automatic options handling.

app.blueprints.users.NewUserResource.representations**NewUserResource.representations** = `None`**app.blueprints.users.NewUserResource.task_service****NewUserResource.task_service** = `<app.services.task.TaskService object>`

app.blueprints.users.NewUserResource.user_serializer

```
NewUserResource.user_serializer = <UserSerializer(many=False)>
```

app.blueprints.users.NewUserResource.user_service

```
NewUserResource.user_service = <app.services.user.UserService object>
```

Methods

<i>NewUserResource.__init__</i> ([api])	
<i>NewUserResource.as_view</i> (name, *class_args, ...)	Converts the class into an actual view function that can be used with the routing system.
<i>NewUserResource.dispatch_request</i> (*args, **kwargs)	Subclasses have to override this method to implement the actual view function code.
<i>NewUserResource.post</i> ()	
<i>NewUserResource.validate_payload</i> (func)	Perform a payload validation on expected model if necessary

app.blueprints.users.NewUserResource.__init__

```
NewUserResource.__init__(api=None, *args, **kwargs)
```

app.blueprints.users.NewUserResource.as_view

classmethod *NewUserResource.as_view*(name: str, *class_args: Any, **class_kwargs: Any)
→ Callable

Converts the class into an actual view function that can be used with the routing system. Internally this generates a function on the fly which will instantiate the *View* on each request and call the *dispatch_request()* method on it.

The arguments passed to *as_view()* are forwarded to the constructor of the class.

app.blueprints.users.NewUserResource.dispatch_request

```
NewUserResource.dispatch_request(*args, **kwargs)
```

Subclasses have to override this method to implement the actual view function code. This method is called with all the arguments from the URL rule.

app.blueprints.users.NewUserResource.post

`NewUserResource.post()` → tuple

app.blueprints.users.NewUserResource.validate_payload

`NewUserResource.validate_payload(func)`
Perform a payload validation on expected model if necessary

app.blueprints.users.UserBaseResource

class `app.blueprints.users.UserBaseResource`(*api=None, *args, **kwargs*)
Bases: `app.blueprints.base.BaseResource`

Attributes

<code>UserBaseResource.decorators</code>	The canonical way to decorate class-based views is to decorate the return value of <code>as_view()</code> .
<code>UserBaseResource.method_decorators</code>	
<code>UserBaseResource.methods</code>	A list of methods this view can handle.
<code>UserBaseResource.provide_automatic_options</code>	Setting this disables or force-enables the automatic options handling.
<code>UserBaseResource.representations</code>	
<code>UserBaseResource.task_service</code>	
<code>UserBaseResource.user_serializer</code>	
<code>UserBaseResource.user_service</code>	

app.blueprints.users.UserBaseResource.decorators

`UserBaseResource.decorators: List[Callable] = []`

The canonical way to decorate class-based views is to decorate the return value of `as_view()`. However since this moves parts of the logic from the class declaration to the place where it's hooked into the routing system.

You can place one or more decorators in this list and whenever the view function is created the result is automatically decorated.

New in version 0.8.

app.blueprints.users.UserBaseResource.method_decorators

```
UserBaseResource.method_decorators = []
```

app.blueprints.users.UserBaseResource.methods

```
UserBaseResource.methods: Optional[List[str]] = None
```

A list of methods this view can handle.

app.blueprints.users.UserBaseResource.provide_automatic_options

```
UserBaseResource.provide_automatic_options: Optional[bool] = None
```

Setting this disables or force-enables the automatic options handling.

app.blueprints.users.UserBaseResource.representations

```
UserBaseResource.representations = None
```

app.blueprints.users.UserBaseResource.task_service

```
UserBaseResource.task_service = <app.services.task.TaskService object>
```

app.blueprints.users.UserBaseResource.user_serializer

```
UserBaseResource.user_serializer = <UserSerializer(many=False)>
```

app.blueprints.users.UserBaseResource.user_service

```
UserBaseResource.user_service = <app.services.user.UserService object>
```

Methods

<code>UserBaseResource.__init__([api])</code>	
<code>UserBaseResource.as_view(name, *class_args, ...)</code>	Converts the class into an actual view function that can be used with the routing system.
<code>UserBaseResource.dispatch_request(*args, ...)</code>	Subclasses have to override this method to implement the actual view function code.
<code>UserBaseResource.validate_payload(func)</code>	Perform a payload validation on expected model if necessary

app.blueprints.users.UserBaseResource.__init__

`UserBaseResource.__init__(api=None, *args, **kwargs)`

app.blueprints.users.UserBaseResource.as_view

classmethod `UserBaseResource.as_view(name: str, *class_args: Any, **class_kwargs: Any) → Callable`

Converts the class into an actual view function that can be used with the routing system. Internally this generates a function on the fly which will instantiate the `View` on each request and call the `dispatch_request()` method on it.

The arguments passed to `as_view()` are forwarded to the constructor of the class.

app.blueprints.users.UserBaseResource.dispatch_request

`UserBaseResource.dispatch_request(*args, **kwargs)`

Subclasses have to override this method to implement the actual view function code. This method is called with all the arguments from the URL rule.

app.blueprints.users.UserBaseResource.validate_payload

`UserBaseResource.validate_payload(func)`

Perform a payload validation on expected model if necessary

app.blueprints.users.UserResource

class `app.blueprints.users.UserResource(api=None, *args, **kwargs)`

Bases: `app.blueprints.users.UserBaseResource`

Attributes

<code>UserResource.decorators</code>	The canonical way to decorate class-based views is to decorate the return value of <code>as_view()</code> .
<code>UserResource.method_decorators</code>	
<code>UserResource.methods</code>	A list of methods this view can handle.
<code>UserResource.provide_automatic_options</code>	Setting this disables or force-enables the automatic options handling.
<code>UserResource.representations</code>	
<code>UserResource.task_service</code>	
<code>UserResource.user_serializer</code>	
<code>UserResource.user_service</code>	

app.blueprints.users.UserResource.decorators

UserResource.decorators: `List[Callable] = []`

The canonical way to decorate class-based views is to decorate the return value of `as_view()`. However since this moves parts of the logic from the class declaration to the place where it's hooked into the routing system.

You can place one or more decorators in this list and whenever the view function is created the result is automatically decorated.

New in version 0.8.

app.blueprints.users.UserResource.method_decorators

UserResource.method_decorators = `[]`

app.blueprints.users.UserResource.methods

UserResource.methods: `Optional[List[str]] = {'DELETE', 'GET', 'PUT'}`

A list of methods this view can handle.

app.blueprints.users.UserResource.provide_automatic_options

UserResource.provide_automatic_options: `Optional[bool] = None`

Setting this disables or force-enables the automatic options handling.

app.blueprints.users.UserResource.representations

UserResource.representations = `None`

app.blueprints.users.UserResource.task_service

UserResource.task_service = `<app.services.task.TaskService object>`

app.blueprints.users.UserResource.user_serializer

UserResource.user_serializer = `<UserSerializer(many=False)>`

app.blueprints.users.UserResource.user_service

UserResource.user_service = `<app.services.user.UserService object>`

Methods

<hr/> <i>UserResource.__init__</i> ([api]) <hr/>	
<i>UserResource.as_view</i> (name, *class_args, ...)	Converts the class into an actual view function that can be used with the routing system.
<hr/> <i>UserResource.delete</i> (user_id) <hr/>	
<i>UserResource.dispatch_request</i> (*args, **kwargs)	Subclasses have to override this method to implement the actual view function code.
<hr/> <i>UserResource.get</i> (user_id) <hr/>	
<hr/> <i>UserResource.put</i> (user_id) <hr/>	
<i>UserResource.validate_payload</i> (func)	Perform a payload validation on expected model if necessary
<hr/>	

app.blueprints.users.UserResource.__init__

`UserResource.__init__(api=None, *args, **kwargs)`

app.blueprints.users.UserResource.as_view

classmethod `UserResource.as_view(name: str, *class_args: Any, **class_kwargs: Any) → Callable`

Converts the class into an actual view function that can be used with the routing system. Internally this generates a function on the fly which will instantiate the `View` on each request and call the `dispatch_request()` method on it.

The arguments passed to `as_view()` are forwarded to the constructor of the class.

app.blueprints.users.UserResource.delete

`UserResource.delete(user_id: int) → tuple`

app.blueprints.users.UserResource.dispatch_request

`UserResource.dispatch_request(*args, **kwargs)`

Subclasses have to override this method to implement the actual view function code. This method is called with all the arguments from the URL rule.

app.blueprints.users.UserResource.get

UserResource.get(*user_id: int*) → tuple

app.blueprints.users.UserResource.put

UserResource.put(*user_id: int*) → tuple

app.blueprints.users.UserResource.validate_payload

UserResource.validate_payload(*func*)
Perform a payload validation on expected model if necessary

app.blueprints.users.UsersSearchResource

class app.blueprints.users.UsersSearchResource(*api=None, *args, **kwargs*)

Bases: *app.blueprints.users.UserBaseResource*

Attributes

<i>UsersSearchResource.decorators</i>	The canonical way to decorate class-based views is to decorate the return value of <code>as_view()</code> .
<i>UsersSearchResource.method_decorators</i>	
<i>UsersSearchResource.methods</i>	A list of methods this view can handle.
<i>UsersSearchResource.provide_automatic_options</i>	Setting this disables or force-enables the automatic options handling.
<i>UsersSearchResource.representations</i>	
<i>UsersSearchResource.task_service</i>	
<i>UsersSearchResource.user_serializer</i>	
<i>UsersSearchResource.user_service</i>	

app.blueprints.users.UsersSearchResource.decorators

UsersSearchResource.decorators: List[Callable] = []

The canonical way to decorate class-based views is to decorate the return value of `as_view()`. However since this moves parts of the logic from the class declaration to the place where it's hooked into the routing system.

You can place one or more decorators in this list and whenever the view function is created the result is automatically decorated.

New in version 0.8.

app.blueprints.users.UsersSearchResource.method_decorators

UsersSearchResource.method_decorators = []

app.blueprints.users.UsersSearchResource.methods

UsersSearchResource.methods: Optional[List[str]] = {'POST'}
A list of methods this view can handle.

app.blueprints.users.UsersSearchResource.provide_automatic_options

UsersSearchResource.provide_automatic_options: Optional[bool] = None
Setting this disables or force-enables the automatic options handling.

app.blueprints.users.UsersSearchResource.representations

UsersSearchResource.representations = None

app.blueprints.users.UsersSearchResource.task_service

UsersSearchResource.task_service = <app.services.task.TaskService object>

app.blueprints.users.UsersSearchResource.user_serializer

UsersSearchResource.user_serializer = <UserSerializer(many=False)>

app.blueprints.users.UsersSearchResource.user_service

UsersSearchResource.user_service = <app.services.user.UserService object>

Methods

<i>UsersSearchResource.__init__([api])</i>	
<i>UsersSearchResource.as_view</i> (name, ...)	Converts the class into an actual view function that can be used with the routing system.
<i>UsersSearchResource.dispatch_request</i> (*args, ...)	Subclasses have to override this method to implement the actual view function code.
<i>UsersSearchResource.post</i> ()	
<i>UsersSearchResource.validate_payload</i> (func)	Perform a payload validation on expected model if necessary

app.blueprints.users.UsersSearchResource.__init__

`UsersSearchResource.__init__(api=None, *args, **kwargs)`

app.blueprints.users.UsersSearchResource.as_view

classmethod `UsersSearchResource.as_view(name: str, *class_args: Any, **class_kwargs: Any) → Callable`

Converts the class into an actual view function that can be used with the routing system. Internally this generates a function on the fly which will instantiate the `View` on each request and call the `dispatch_request()` method on it.

The arguments passed to `as_view()` are forwarded to the constructor of the class.

app.blueprints.users.UsersSearchResource.dispatch_request

`UsersSearchResource.dispatch_request(*args, **kwargs)`

Subclasses have to override this method to implement the actual view function code. This method is called with all the arguments from the URL rule.

app.blueprints.users.UsersSearchResource.post

`UsersSearchResource.post() → tuple`

app.blueprints.users.UsersSearchResource.validate_payload

`UsersSearchResource.validate_payload(func)`

Perform a payload validation on expected model if necessary

class `app.blueprints.users.ExportUsersExcelAndWordResource(api=None, *args, **kwargs)`

classmethod `as_view(name: str, *class_args: Any, **class_kwargs: Any) → Callable`

Converts the class into an actual view function that can be used with the routing system. Internally this generates a function on the fly which will instantiate the `View` on each request and call the `dispatch_request()` method on it.

The arguments passed to `as_view()` are forwarded to the constructor of the class.

decorators: `List[Callable] = []`

The canonical way to decorate class-based views is to decorate the return value of `as_view()`. However since this moves parts of the logic from the class declaration to the place where it's hooked into the routing system.

You can place one or more decorators in this list and whenever the view function is created the result is automatically decorated.

New in version 0.8.

dispatch_request `(*args, **kwargs)`

Subclasses have to override this method to implement the actual view function code. This method is called with all the arguments from the URL rule.

method_decorators `= []`

methods: `Optional[List[str]] = {'POST'}`

A list of methods this view can handle.

post() \rightarrow tuple

provide_automatic_options: `Optional[bool] = None`

Setting this disables or force-enables the automatic options handling.

representations = `None`

task_service = `<app.services.task.TaskService object>`

user_serializer = `<UserSerializer(many=False)>`

user_service = `<app.services.user.UserService object>`

validate_payload(*func*)

Perform a payload validation on expected model if necessary

class `app.blueprints.users.ExportUsersExcelResource`(*api=None, *args, **kwargs*)

classmethod `as_view`(*name: str, *class_args: Any, **class_kwargs: Any*) \rightarrow Callable

Converts the class into an actual view function that can be used with the routing system. Internally this generates a function on the fly which will instantiate the View on each request and call the `dispatch_request()` method on it.

The arguments passed to `as_view()` are forwarded to the constructor of the class.

decorators: `List[Callable] = []`

The canonical way to decorate class-based views is to decorate the return value of `as_view()`. However since this moves parts of the logic from the class declaration to the place where it's hooked into the routing system.

You can place one or more decorators in this list and whenever the view function is created the result is automatically decorated.

New in version 0.8.

dispatch_request(**args, **kwargs*)

Subclasses have to override this method to implement the actual view function code. This method is called with all the arguments from the URL rule.

method_decorators = `[]`

methods: `Optional[List[str]] = {'POST'}`

A list of methods this view can handle.

post() \rightarrow tuple

provide_automatic_options: `Optional[bool] = None`

Setting this disables or force-enables the automatic options handling.

representations = `None`

task_service = `<app.services.task.TaskService object>`

user_serializer = `<UserSerializer(many=False)>`

user_service = `<app.services.user.UserService object>`

validate_payload(*func*)

Perform a payload validation on expected model if necessary

class `app.blueprints.users.ExportUsersWordResource`(*api=None, *args, **kwargs*)

classmethod as_view(*name: str, *class_args: Any, **class_kwargs: Any*) → Callable

Converts the class into an actual view function that can be used with the routing system. Internally this generates a function on the fly which will instantiate the View on each request and call the `dispatch_request()` method on it.

The arguments passed to `as_view()` are forwarded to the constructor of the class.

decorators: List[Callable] = []

The canonical way to decorate class-based views is to decorate the return value of `as_view()`. However since this moves parts of the logic from the class declaration to the place where it's hooked into the routing system.

You can place one or more decorators in this list and whenever the view function is created the result is automatically decorated.

New in version 0.8.

dispatch_request(**args, **kwargs*)

Subclasses have to override this method to implement the actual view function code. This method is called with all the arguments from the URL rule.

method_decorators = []

methods: Optional[List[str]] = {'POST'}

A list of methods this view can handle.

post() → tuple

provide_automatic_options: Optional[bool] = None

Setting this disables or force-enables the automatic options handling.

representations = None

task_service = <app.services.task.TaskService object>

user_serializer = <UserSerializer(many=False)>

user_service = <app.services.user.UserService object>

validate_payload(*func*)

Perform a payload validation on expected model if necessary

class app.blueprints.users.NewUserResource(*api=None, *args, **kwargs*)

classmethod as_view(*name: str, *class_args: Any, **class_kwargs: Any*) → Callable

Converts the class into an actual view function that can be used with the routing system. Internally this generates a function on the fly which will instantiate the View on each request and call the `dispatch_request()` method on it.

The arguments passed to `as_view()` are forwarded to the constructor of the class.

decorators: List[Callable] = []

The canonical way to decorate class-based views is to decorate the return value of `as_view()`. However since this moves parts of the logic from the class declaration to the place where it's hooked into the routing system.

You can place one or more decorators in this list and whenever the view function is created the result is automatically decorated.

New in version 0.8.

dispatch_request(*args, **kwargs)

Subclasses have to override this method to implement the actual view function code. This method is called with all the arguments from the URL rule.

method_decorators = []

methods: Optional[List[str]] = {'POST'}

A list of methods this view can handle.

post() → tuple

provide_automatic_options: Optional[bool] = None

Setting this disables or force-enables the automatic options handling.

representations = None

task_service = <app.services.task.TaskService object>

user_serializer = <UserSerializer(many=False)>

user_service = <app.services.user.UserService object>

validate_payload(func)

Perform a payload validation on expected model if necessary

class app.blueprints.users.UserBaseResource(*api=None, *args, **kwargs*)

classmethod **as_view**(name: str, *class_args: Any, **class_kwargs: Any) → Callable

Converts the class into an actual view function that can be used with the routing system. Internally this generates a function on the fly which will instantiate the View on each request and call the [dispatch_request\(\)](#) method on it.

The arguments passed to [as_view\(\)](#) are forwarded to the constructor of the class.

decorators: List[Callable] = []

The canonical way to decorate class-based views is to decorate the return value of [as_view\(\)](#). However since this moves parts of the logic from the class declaration to the place where it's hooked into the routing system.

You can place one or more decorators in this list and whenever the view function is created the result is automatically decorated.

New in version 0.8.

dispatch_request(*args, **kwargs)

Subclasses have to override this method to implement the actual view function code. This method is called with all the arguments from the URL rule.

method_decorators = []

methods: Optional[List[str]] = None

A list of methods this view can handle.

provide_automatic_options: Optional[bool] = None

Setting this disables or force-enables the automatic options handling.

representations = None

task_service = <app.services.task.TaskService object>

user_serializer = <UserSerializer(many=False)>

user_service = <app.services.user.UserService object>

validate_payload(func)

Perform a payload validation on expected model if necessary

class app.blueprints.users.**UserResource**(api=None, *args, **kwargs)

classmethod as_view(name: str, *class_args: Any, **class_kwargs: Any) → Callable

Converts the class into an actual view function that can be used with the routing system. Internally this generates a function on the fly which will instantiate the View on each request and call the `dispatch_request()` method on it.

The arguments passed to `as_view()` are forwarded to the constructor of the class.

decorators: List[Callable] = []

The canonical way to decorate class-based views is to decorate the return value of `as_view()`. However since this moves parts of the logic from the class declaration to the place where it's hooked into the routing system.

You can place one or more decorators in this list and whenever the view function is created the result is automatically decorated.

New in version 0.8.

delete(user_id: int) → tuple

dispatch_request(*args, **kwargs)

Subclasses have to override this method to implement the actual view function code. This method is called with all the arguments from the URL rule.

get(user_id: int) → tuple

method_decorators = []

methods: Optional[List[str]] = {'DELETE', 'GET', 'PUT'}

A list of methods this view can handle.

provide_automatic_options: Optional[bool] = None

Setting this disables or force-enables the automatic options handling.

put(user_id: int) → tuple

representations = None

task_service = <app.services.task.TaskService object>

user_serializer = <UserSerializer(many=False)>

user_service = <app.services.user.UserService object>

validate_payload(func)

Perform a payload validation on expected model if necessary

class app.blueprints.users.**UsersSearchResource**(api=None, *args, **kwargs)

classmethod as_view(name: str, *class_args: Any, **class_kwargs: Any) → Callable

Converts the class into an actual view function that can be used with the routing system. Internally this generates a function on the fly which will instantiate the View on each request and call the `dispatch_request()` method on it.

The arguments passed to `as_view()` are forwarded to the constructor of the class.

decorators: List[Callable] = []

The canonical way to decorate class-based views is to decorate the return value of `as_view()`. However

since this moves parts of the logic from the class declaration to the place where it's hooked into the routing system.

You can place one or more decorators in this list and whenever the view function is created the result is automatically decorated.

New in version 0.8.

dispatch_request(*args, **kwargs)

Subclasses have to override this method to implement the actual view function code. This method is called with all the arguments from the URL rule.

method_decorators = []

methods: Optional[List[str]] = {'POST'}

A list of methods this view can handle.

post() → tuple

provide_automatic_options: Optional[bool] = None

Setting this disables or force-enables the automatic options handling.

representations = None

task_service = <app.services.task.TaskService object>

user_serializer = <UserSerializer(many=False)>

user_service = <app.services.user.UserService object>

validate_payload(func)

Perform a payload validation on expected model if necessary

Functions

[*get_blueprints\(\)*](#)

Get Blueprints via dynamic way.

app.blueprints.get_blueprints

app.blueprints.get_blueprints() → list

Get Blueprints via dynamic way.

app.blueprints.get_blueprints() → list

Get Blueprints via dynamic way.

2.1.2 app.celery

Description

Runs Celery and registers Celery tasks.

Modules

app.celery.excel

app.celery.tasks

app.celery.tests

app.celery.word

app.celery.excel

Description

Modules

app.celery.excel.tasks

app.celery.excel.tasks

Description

`app.celery.excel.tasks._add_excel_autofilter`(*worksheet: xlsxwriter.worksheet.Worksheet*)

`app.celery.excel.tasks._adjust_each_column_width`(*rows: list, worksheet: xlsxwriter.worksheet.Worksheet, excel_longest_word: int*) → None

`app.celery.excel.tasks._get_excel_column_names`(*excel_rows: list*) → None

`app.celery.excel.tasks._get_excel_user_data`(*users: list, excel_rows: list*) → None

`app.celery.excel.tasks._get_user_data`(*request_data: dict*) → list

`app.celery.excel.tasks._parse_user_data`(*users: list*)

(task) `app.celery.excel.tasks.export_user_data_in_excel_task`(*created_by: int, request_data: dict*)

Proxy that evaluates object once.

Proxy will evaluate the object each time, while the promise will only evaluate it once.

app.celery.tasks

Description

(task) `app.celery.tasks.create_user_email_task(email_data)` → bool

Proxy that evaluates object once.

Proxy will evaluate the object each time, while the promise will only evaluate it once.

(task) `app.celery.tasks.create_word_and_excel_documents_task(created_by: int, request_data: dict, to_pdf: int)` → bool

Proxy that evaluates object once.

Proxy will evaluate the object each time, while the promise will only evaluate it once.

(task) `app.celery.tasks.reset_password_email_task(email_data)` → bool

Proxy that evaluates object once.

Proxy will evaluate the object each time, while the promise will only evaluate it once.

(task) `app.celery.tasks.send_email_with_attachments_task(task_data: list)` → bool

Proxy that evaluates object once.

Proxy will evaluate the object each time, while the promise will only evaluate it once.

app.celery.tests

Description

Modules

app.celery.tests.tasks

app.celery.tests.tasks

Description

(task) `app.celery.tests.tasks.create_task_table()` → None

Proxy that evaluates object once.

Proxy will evaluate the object each time, while the promise will only evaluate it once.

(task) `app.celery.tests.tasks.insert_task_record()` → None

Proxy that evaluates object once.

Proxy will evaluate the object each time, while the promise will only evaluate it once.

app.celery.word

Description

Modules

app.celery.word.tasks

app.celery.word.tasks

Description

`app.celery.word.tasks._add_table_column_names(rows: list, original_column_names: set) → None`

`app.celery.word.tasks._add_table_user_data(users_query: list, rows: list) → None`

`app.celery.word.tasks._get_user_data(request_data: dict) → list`

(task)`app.celery.word.tasks.export_user_data_in_word_task(created_by: int, request_data: dict, to_pdf: int)`

Proxy that evaluates object once.

Proxy will evaluate the object each time, while the promise will only evaluate it once.

Classes

ContextTask()

MyCelery([main, loader, backend, amqp, ...])

app.celery.ContextTask

class `app.celery.ContextTask`
Bases: `celery.app.task.Task`

Attributes

<i>ContextTask.Request</i>	Request class used, or the qualified name of one.
<i>ContextTask.Strategy</i>	Execution strategy used, or the qualified name of one.
<i>ContextTask.abstract</i>	Deprecated attribute <code>abstract</code> here for compatibility.
<i>ContextTask.acks_late</i>	When enabled messages for this task will be acknowledged after the task has been executed, and not <i>just before</i> (the default behavior).

continues on next page

Table 64 – continued from previous page

<code>ContextTask.acks_on_failure_or_timeout</code>	When enabled messages for this task will be acknowledged even if it fails or times out.
<code>ContextTask.app</code>	
<code>ContextTask.backend</code>	
<code>ContextTask.default_retry_delay</code>	Default time in seconds before a retry of the task should be executed.
<code>ContextTask.expires</code>	Default task expiry time.
<code>ContextTask.from_config</code>	
<code>ContextTask.ignore_result</code>	If enabled the worker won't store task state and return values for this task. Defaults to the :setting:task_ignore_result setting.
<code>ContextTask.max_retries</code>	Maximum number of retries before giving up.
<code>ContextTask.name</code>	Name of the task.
<code>ContextTask.priority</code>	Default task priority.
<code>ContextTask.rate_limit</code>	None (no rate limit), <code>'100/s'</code> (hundred tasks a second), <code>'100/m'</code> (hundred tasks a minute), <code>'100/h'</code> (hundred tasks an hour)
<code>ContextTask.reject_on_worker_lost</code>	Even if <code>acks_late</code> is enabled, the worker will acknowledge tasks when the worker process executing them abruptly exits or is signaled (e.g., :sig:KILL / :sig:INT , etc).
<code>ContextTask.request</code>	Get current request object.
<code>ContextTask.request_stack</code>	Task request stack, the current request will be the topmost.
<code>ContextTask.resultrepr_maxsize</code>	Max length of result representation used in logs and events.
<code>ContextTask.send_events</code>	If enabled the worker will send monitoring events related to this task (but only if the worker is configured to send task related events).
<code>ContextTask.serializer</code>	The name of a serializer that are registered with <code>kombu.serialization.registry</code> .
<code>ContextTask.soft_time_limit</code>	Soft time limit. Defaults to the :setting:task_soft_time_limit setting.
<code>ContextTask.store_eager_result</code>	
<code>ContextTask.store_errors_even_if_ignored</code>	When enabled errors will be stored even if the task is otherwise configured to ignore results.
<code>ContextTask.raises</code>	Tuple of expected exceptions.
<code>ContextTask.time_limit</code>	Hard time limit. Defaults to the :setting:task_time_limit setting.
<code>ContextTask.track_started</code>	If enabled the task will report its status as 'started' when the task is executed by a worker.
<code>ContextTask.trail</code>	If enabled the request will keep track of sub-tasks started by this task, and this information will be sent with the result (<code>result.children</code>).
<code>ContextTask.typing</code>	Enable argument checking.

`app.celery.ContextTask.Request`

`ContextTask.Request = 'celery.worker.request:Request'`

Request class used, or the qualified name of one.

`app.celery.ContextTask.Strategy`

`ContextTask.Strategy = 'celery.worker.strategy:default'`

Execution strategy used, or the qualified name of one.

`app.celery.ContextTask.abstract`

`ContextTask.abstract = True`

Deprecated attribute `abstract` here for compatibility.

`app.celery.ContextTask.acks_late`

`ContextTask.acks_late = False`

When enabled messages for this task will be acknowledged **after** the task has been executed, and not *just before* (the default behavior).

Please note that this means the task may be executed twice if the worker crashes mid execution.

The application default can be overridden with the `:setting:`task_acks_late`` setting.

`app.celery.ContextTask.acks_on_failure_or_timeout`

`ContextTask.acks_on_failure_or_timeout = True`

When enabled messages for this task will be acknowledged even if it fails or times out.

Configuring this setting only applies to tasks that are acknowledged **after** they have been executed and only if `:setting:`task_acks_late`` is enabled.

The application default can be overridden with the `:setting:`task_acks_on_failure_or_timeout`` setting.

`app.celery.ContextTask.app`

`ContextTask.app = <MyCelery __main__>`

`app.celery.ContextTask.backend`

`property ContextTask.backend`

app.celery.ContextTask.default_retry_delay

ContextTask.default_retry_delay = 180

Default time in seconds before a retry of the task should be executed. 3 minutes by default.

app.celery.ContextTask.expires

ContextTask.expires = None

Default task expiry time.

app.celery.ContextTask.from_config

```
ContextTask.from_config = (('serializer', 'task_serializer'), ('rate_limit',
'task_default_rate_limit'), ('priority', 'task_default_priority'),
('track_started', 'task_track_started'), ('acks_late', 'task_acks_late'),
('acks_on_failure_or_timeout', 'task_acks_on_failure_or_timeout'),
('reject_on_worker_lost', 'task_reject_on_worker_lost'), ('ignore_result',
'task_ignore_result'), ('store_eager_result', 'task_store_eager_result'),
('store_errors_even_if_ignored', 'task_store_errors_even_if_ignored'))
```

app.celery.ContextTask.ignore_result

ContextTask.ignore_result = False

If enabled the worker won't store task state and return values for this task. Defaults to the **:setting:task_ignore_result** setting.

app.celery.ContextTask.max_retries

ContextTask.max_retries = 3

Maximum number of retries before giving up. If set to None, it will **never** stop retrying.

app.celery.ContextTask.name

ContextTask.name = None

Name of the task.

app.celery.ContextTask.priority

ContextTask.priority = None

Default task priority.

app.celery.ContextTask.rate_limit**ContextTask.rate_limit = None**

None (no rate limit), `'100/s'` (hundred tasks a second), `'100/m'` (hundred tasks a minute), `'100/h'` (hundred tasks an hour)

Type Rate limit for this task type. Examples

app.celery.ContextTask.reject_on_worker_lost**ContextTask.reject_on_worker_lost = None**

Even if `acks_late` is enabled, the worker will acknowledge tasks when the worker process executing them abruptly exits or is signaled (e.g., `:sig:KILL`/`:sig:INT`, etc).

Setting this to true allows the message to be re-queued instead, so that the task will execute again by the same worker, or another worker.

Warning: Enabling this can cause message loops; make sure you know what you're doing.

app.celery.ContextTask.request**property ContextTask.request**

Get current request object.

app.celery.ContextTask.request_stack**ContextTask.request_stack = <celery.utils.threads._LocalStack object>**

Task request stack, the current request will be the topmost.

app.celery.ContextTask.resultrepr_maxsize**ContextTask.resultrepr_maxsize = 1024**

Max length of result representation used in logs and events.

app.celery.ContextTask.send_events**ContextTask.send_events = True**

If enabled the worker will send monitoring events related to this task (but only if the worker is configured to send task related events). Note that this has no effect on the task-failure event case where a task is not registered (as it will have no task class to check this flag).

app.celery.ContextTask.serializer

ContextTask.**serializer** = 'json'

The name of a serializer that are registered with kombu.serialization.registry. Default is 'json'.

app.celery.ContextTask.soft_time_limit

ContextTask.**soft_time_limit** = None

Soft time limit. Defaults to the **:setting:`task_soft_time_limit`** setting.

app.celery.ContextTask.store_eager_result

ContextTask.**store_eager_result** = False

app.celery.ContextTask.store_errors_even_if_ignored

ContextTask.**store_errors_even_if_ignored** = False

When enabled errors will be stored even if the task is otherwise configured to ignore results.

app.celery.ContextTask.raises

ContextTask.**raises** = ()

Tuple of expected exceptions.

These are errors that are expected in normal operation and that shouldn't be regarded as a real error by the worker. Currently this means that the state will be updated to an error state, but the worker won't log the event as an error.

app.celery.ContextTask.time_limit

ContextTask.**time_limit** = None

Hard time limit. Defaults to the **:setting:`task_time_limit`** setting.

app.celery.ContextTask.track_started

ContextTask.**track_started** = False

If enabled the task will report its status as 'started' when the task is executed by a worker. Disabled by default as the normal behavior is to not report that level of granularity. Tasks are either pending, finished, or waiting to be retried.

Having a 'started' status can be useful for when there are long running tasks and there's a need to report what task is currently running.

The application default can be overridden using the **:setting:`task_track_started`** setting.

app.celery.ContextTask.trail**ContextTask.trail = True**

If enabled the request will keep track of subtasks started by this task, and this information will be sent with the result (`result.children`).

app.celery.ContextTask.typing**ContextTask.typing = True**

Enable argument checking. You can set this to false if you don't want the signature to be checked when calling the task. Defaults to `app.strict_typing`.

Methods

<code>ContextTask.AsyncResult(task_id, **kwargs)</code>	Get AsyncResult instance for the specified task.
<code>ContextTask.__init__()</code>	
<code>ContextTask.add_around(attr, around)</code>	
<code>ContextTask.add_to_chord(sig[, lazy])</code>	Add signature to the chord the current task is a member of.
<code>ContextTask.add_trail(result)</code>	
<code>ContextTask.after_return(status, retval, ...)</code>	Handler called after the task returns.
<code>ContextTask.annotate()</code>	
<code>ContextTask.apply([args, kwargs, link, ...])</code>	Execute this task locally, by blocking until the task returns.
<code>ContextTask.apply_async([args, kwargs, ...])</code>	Apply tasks asynchronously by sending a message.
<code>ContextTask.before_start(task_id, args, kwargs)</code>	Handler called before the task starts.
<code>ContextTask.bind(app)</code>	
<code>ContextTask.chunks(it, n)</code>	Create a <code>chunks</code> task for this task.
<code>ContextTask.delay(*args, **kwargs)</code>	Star argument version of <code>apply_async()</code> .
<code>ContextTask.map(it)</code>	Create a <code>xmap</code> task from <code>it</code> .
<code>ContextTask.on_bound(app)</code>	Called when the task is bound to an app.
<code>ContextTask.on_failure(exc, task_id, args, ...)</code>	Error handler.
<code>ContextTask.on_retry(exc, task_id, args, ...)</code>	Retry handler.
<code>ContextTask.on_success(retval, task_id, ...)</code>	Success handler.
<code>ContextTask.pop_request()</code>	
<code>ContextTask.push_request(*args, **kwargs)</code>	

continues on next page

Table 65 – continued from previous page

<code>ContextTask.replace(sig)</code>	Replace this task, with a new task inheriting the task id.
<code>ContextTask.retry([args, kwargs, exc, ...])</code>	Retry the task, adding it to the back of the queue.
<code>ContextTask.run(*args, **kwargs)</code>	The body of the task executed by workers.
<code>ContextTask.s(*args, **kwargs)</code>	Create signature.
<code>ContextTask.send_event(type_[, retry, ...])</code>	Send monitoring event message.
<code>ContextTask.shadow_name(args, kwargs, options)</code>	Override for custom task name in worker logs/monitoring.
<code>ContextTask.si(*args, **kwargs)</code>	Create immutable signature.
<code>ContextTask.signature([args])</code>	Create signature.
<code>ContextTask. signature_from_request([...])</code>	
<code>ContextTask.starmap(it)</code>	Create a <code>xstarmap</code> task from it.
<code>ContextTask.start_strategy(app, consumer, ...)</code>	
<code>ContextTask.subtask([args])</code>	Create signature.
<code>ContextTask. subtask_from_request([request, ...])</code>	
<code>ContextTask.update_state([task_id, state, meta])</code>	Update task state.

app.celery.ContextTask.AsyncResult

`ContextTask.AsyncResult(task_id, **kwargs)`

Get AsyncResult instance for the specified task.

Parameters `task_id` (*str*) – Task id to get result for.

app.celery.ContextTask.__init__

`ContextTask.__init__()`

app.celery.ContextTask.add_around

classmethod `ContextTask.add_around(attr, around)`

app.celery.ContextTask.add_to_chord

`ContextTask.add_to_chord(sig, lazy=False)`

Add signature to the chord the current task is a member of.

New in version 4.0.

Currently only supported by the Redis result backend.

Parameters

- **sig** (*Signature*) – Signature to extend chord with.
- **lazy** (*bool*) – If enabled the new task won't actually be called, and `sig.delay()` must be called manually.

app.celery.ContextTask.add_trail

ContextTask.**add_trail**(*result*)

app.celery.ContextTask.after_return

ContextTask.**after_return**(*status, retval, task_id, args, kwargs, einfo*)

Handler called after the task returns.

Parameters

- **status** (*str*) – Current task state.
- **retval** (*Any*) – Task return value/exception.
- **task_id** (*str*) – Unique id of the task.
- **args** (*Tuple*) – Original arguments for the task.
- **kwargs** (*Dict*) – Original keyword arguments for the task.
- **einfo** (*ExceptionInfo*) – Exception information.

Returns The return value of this handler is ignored.

Return type None

app.celery.ContextTask.annotate

classmethod ContextTask.**annotate**()

app.celery.ContextTask.apply

ContextTask.**apply**(*args=None, kwargs=None, link=None, link_error=None, task_id=None, retries=None, throw=None, logfile=None, loglevel=None, headers=None, **options*)

Execute this task locally, by blocking until the task returns.

Parameters

- **args** (*Tuple*) – positional arguments passed on to the task.
- **kwargs** (*Dict*) – keyword arguments passed on to the task.
- **throw** (*bool*) – Re-raise task exceptions. Defaults to the **:setting:task_eager_propagates** setting.

Returns pre-evaluated result.

Return type celery.result.EagerResult

app.celery.ContextTask.apply_async

ContextTask.**apply_async**(*args=None, kwargs=None, task_id=None, producer=None, link=None, link_error=None, shadow=None, **options*)

Apply tasks asynchronously by sending a message.

Parameters

- **args** (*Tuple*) – The positional arguments to pass on to the task.
- **kwargs** (*Dict*) – The keyword arguments to pass on to the task.
- **countdown** (*float*) – Number of seconds into the future that the task should execute. Defaults to immediate execution.
- **eta** (*datetime*) – Absolute time and date of when the task should be executed. May not be specified if *countdown* is also supplied.
- **expires** (*float, datetime*) – Datetime or seconds in the future for the task should expire. The task won't be executed after the expiration time.

- **shadow** (*str*) – Override task name used in logs/monitoring. Default is retrieved from `shadow_name()`.
- **connection** (*kombu.Connection*) – Re-use existing broker connection instead of acquiring one from the connection pool.
- **retry** (*bool*) – If enabled sending of the task message will be retried in the event of connection loss or failure. Default is taken from the `:setting:`task_publish_retry`` setting. Note that you need to handle the producer/connection manually for this to work.
- **retry_policy** (*Mapping*) – Override the retry policy used. See the `:setting:`task_publish_retry_policy`` setting.
- **time_limit** (*int*) – If set, overrides the default time limit.
- **soft_time_limit** (*int*) – If set, overrides the default soft time limit.
- **queue** (*str*, *kombu.Queue*) – The queue to route the task to. This must be a key present in `:setting:`task_queues``, or `:setting:`task_create_missing_queues`` must be enabled. See guide-routing for more information.
- **exchange** (*str*, *kombu.Exchange*) – Named custom exchange to send the task to. Usually not used in combination with the `queue` argument.
- **routing_key** (*str*) – Custom routing key used to route the task to a worker server. If in combination with a `queue` argument only used to specify custom routing keys to topic exchanges.
- **priority** (*int*) – The task priority, a number between 0 and 9. Defaults to the `priority` attribute.
- **serializer** (*str*) – Serialization method to use. Can be `pickle`, `json`, `yaml`, `msgpack` or any custom serialization method that's been registered with `kombu.serialization.registry`. Defaults to the `serializer` attribute.
- **compression** (*str*) – Optional compression method to use. Can be one of `zlib`, `bzip2`, or any custom compression methods registered with `kombu.compression.register()`. Defaults to the `:setting:`task_compression`` setting.
- **link** (*Signature*) – A single, or a list of tasks signatures to apply if the task returns successfully.
- **link_error** (*Signature*) – A single, or a list of task signatures to apply if an error occurs while executing the task.
- **producer** (*kombu.Producer*) – custom producer to use when publishing the task.
- **add_to_parent** (*bool*) – If set to `True` (default) and the task is applied while executing another task, then the result will be appended to the parent tasks `request.children` attribute. Trailing can also be disabled by default using the `trail` attribute
- **ignore_result** (*bool*) – If set to `False` (default) the result of a task will be stored in the backend. If set to `True` the result will not be stored. This can also be set using the `ignore_result` in the `app.task` decorator.
- **publisher** (*kombu.Producer*) – Deprecated alias to `producer`.
- **headers** (*Dict*) – Message headers to be included in the message.

Returns Promise of future evaluation.

Return type `celery.result.AsyncResult`

Raises

- **TypeError** – If not enough arguments are passed, or too many arguments are passed. Note that signature checks may be disabled by specifying `@task(typing=False)`.
- **kombu.exceptions.OperationalError** – If a connection to the transport cannot be made, or if the connection is lost.

Note: Also supports all keyword arguments supported by `kombu.Producer.publish()`.

app.celery.ContextTask.before_start

`ContextTask.before_start(task_id, args, kwargs)`

Handler called before the task starts.

New in version 5.2.

Parameters

- **task_id** (*str*) – Unique id of the task to execute.
- **args** (*Tuple*) – Original arguments for the task to execute.
- **kwargs** (*Dict*) – Original keyword arguments for the task to execute.

Returns The return value of this handler is ignored.

Return type None

app.celery.ContextTask.bind

classmethod `ContextTask.bind(app)`

app.celery.ContextTask.chunks

`ContextTask.chunks(it, n)`

Create a chunks task for this task.

app.celery.ContextTask.delay

`ContextTask.delay(*args, **kwargs)`

Star argument version of [`apply_async\(\)`](#).

Does not support the extra options enabled by [`apply_async\(\)`](#).

Parameters

- ***args** (*Any*) – Positional arguments passed on to the task.
- ****kwargs** (*Any*) – Keyword arguments passed on to the task.

Returns Future promise.

Return type `celery.result.AsyncResult`

app.celery.ContextTask.map

`ContextTask.map(it)`

Create a `xmap` task from `it`.

app.celery.ContextTask.on_bound

classmethod `ContextTask.on_bound(app)`

Called when the task is bound to an app.

Note: This class method can be defined to do additional actions when the task class is bound to an app.

app.celery.ContextTask.on_failure

`ContextTask.on_failure(exc, task_id, args, kwargs, einfo) → None`
Error handler.

This is run by the worker when the task fails.

Parameters

- **exc** (*Exception*) – The exception raised by the task.
- **task_id** (*str*) – Unique id of the failed task.
- **args** (*Tuple*) – Original arguments for the task that failed.
- **kwargs** (*Dict*) – Original keyword arguments for the task that failed.
- **einfo** (*ExceptionInfo*) – Exception information.

Returns The return value of this handler is ignored.

Return type None

app.celery.ContextTask.on_retry

`ContextTask.on_retry(exc, task_id, args, kwargs, einfo)`
Retry handler.

This is run by the worker when the task is to be retried.

Parameters

- **exc** (*Exception*) – The exception sent to `retry()`.
- **task_id** (*str*) – Unique id of the retried task.
- **args** (*Tuple*) – Original arguments for the retried task.
- **kwargs** (*Dict*) – Original keyword arguments for the retried task.
- **einfo** (*ExceptionInfo*) – Exception information.

Returns The return value of this handler is ignored.

Return type None

app.celery.ContextTask.on_success

`ContextTask.on_success(retval, task_id, args, kwargs)`
Success handler.

Run by the worker if the task executes successfully.

Parameters

- **retval** (*Any*) – The return value of the task.
- **task_id** (*str*) – Unique id of the executed task.
- **args** (*Tuple*) – Original arguments for the executed task.
- **kwargs** (*Dict*) – Original keyword arguments for the executed task.

Returns The return value of this handler is ignored.

Return type None

app.celery.ContextTask.pop_request

ContextTask.**pop_request**()

app.celery.ContextTask.push_request

ContextTask.**push_request**(*args, **kwargs)

app.celery.ContextTask.replace

ContextTask.**replace**(sig)

Replace this task, with a new task inheriting the task id.

Execution of the host task ends immediately and no subsequent statements will be run.

New in version 4.0.

Parameters *sig* (*Signature*) – signature to replace with.

Raises

- **~@Ignore** – This is always raised when called in asynchronous context.
- **It is best to always use return self.replace(...) to convey –**
- **to the reader that the task won't continue after being replaced. –**

app.celery.ContextTask.retry

ContextTask.**retry**(args=None, kwargs=None, exc=None, throw=True, eta=None, countdown=None, max_retries=None, **options)

Retry the task, adding it to the back of the queue.

Example

```
>>> from imaginary_twitter_lib import Twitter
>>> from proj.celery import app
```

```
>>> @app.task(bind=True)
... def tweet(self, auth, message):
...     twitter = Twitter(oauth=auth)
...     try:
...         twitter.post_status_update(message)
...     except twitter.FailWhale as exc:
...         # Retry in 5 minutes.
...         self.retry(countdown=60 * 5, exc=exc)
```

Note: Although the task will never return above as *retry* raises an exception to notify the worker, we use *raise* in front of the *retry* to convey that the rest of the block won't be executed.

Parameters

- **args** (*Tuple*) – Positional arguments to retry with.
- **kwargs** (*Dict*) – Keyword arguments to retry with.

- **exc** (*Exception*) – Custom exception to report when the max retry limit has been exceeded (default: `@MaxRetriesExceededError`).

If this argument is set and retry is called while an exception was raised (`sys.exc_info()` is set) it will attempt to re-raise the current exception.

If no exception was raised it will raise the **exc** argument provided.

- **countdown** (*float*) – Time in seconds to delay the retry for.
- **eta** (*datetime*) – Explicit time and date to run the retry at.
- **max_retries** (*int*) – If set, overrides the default retry limit for this execution. Changes to this parameter don't propagate to subsequent task retry attempts. A value of `None`, means “use the default”, so if you want infinite retries you'd have to set the `max_retries` attribute of the task to `None` first.
- **time_limit** (*int*) – If set, overrides the default time limit.
- **soft_time_limit** (*int*) – If set, overrides the default soft time limit.
- **throw** (*bool*) – If this is `False`, don't raise the `@Retry` exception, that tells the worker to mark the task as being retried. Note that this means the task will be marked as failed if the task raises an exception, or successful if it returns after the retry call.
- ****options** (*Any*) – Extra options to pass on to `apply_async()`.

Raises `celery.exceptions.Retry` – To tell the worker that the task has been re-sent for retry. This always happens, unless the `throw` keyword argument has been explicitly set to `False`, and is considered normal operation.

`app.celery.ContextTask.run`

`ContextTask.run(*args, **kwargs)`

The body of the task executed by workers.

`app.celery.ContextTask.s`

`ContextTask.s(*args, **kwargs)`

Create signature.

Shortcut for `.s(*a, **k) -> .signature(a, k)`.

`app.celery.ContextTask.send_event`

`ContextTask.send_event(type_, retry=True, retry_policy=None, **fields)`

Send monitoring event message.

This can be used to add custom event types in `:pypi:Flower`` and other monitors.

Parameters `type` (*str*) – Type of event, e.g. "task-failed".

Keyword Arguments

- **retry** (*bool*) – Retry sending the message if the connection is lost. Default is taken from the `:setting:task_publish_retry`` setting.
- **retry_policy** (*Mapping*) – Retry settings. Default is taken from the `:setting:task_publish_retry_policy`` setting.
- ****fields** (*Any*) – Map containing information about the event. Must be JSON serializable.

app.celery.ContextTask.shadow_name

ContextTask.**shadow_name**(args, kwargs, options)

Override for custom task name in worker logs/monitoring.

Example

```
from celery.utils.imports import qualname

def shadow_name(task, args, kwargs, options):
    return qualname(args[0])

@app.task(shadow_name=shadow_name, serializer='pickle')
def apply_function_async(fun, *args, **kwargs):
    return fun(*args, **kwargs)
```

Parameters

- **args** (*Tuple*) – Task positional arguments.
- **kwargs** (*Dict*) – Task keyword arguments.
- **options** (*Dict*) – Task execution options.

app.celery.ContextTask.si

ContextTask.**si**(*args, **kwargs)

Create immutable signature.

Shortcut for `.si(*a, **k) -> .signature(a, k, immutable=True)`.

app.celery.ContextTask.signature

ContextTask.**signature**(args=None, *starargs, **starkwargs)

Create signature.

Returns

object for this task, wrapping arguments and execution options for a single task invocation.

Return type signature

app.celery.ContextTask.signature_from_request

ContextTask.**signature_from_request**(request=None, args=None, kwargs=None, queue=None, **extra_options)

app.celery.ContextTask.starmap

ContextTask.**starmap**(*it*)
Create a xstarmap task from it.

app.celery.ContextTask.start_strategy

ContextTask.**start_strategy**(*app, consumer, **kwargs*)

app.celery.ContextTask.subtask

ContextTask.**subtask**(*args=None, *starargs, **starkwargs*)
Create signature.

Returns

object for this task, wrapping arguments and execution options for a single task invocation.

Return type signature

app.celery.ContextTask.subtask_from_request

ContextTask.**subtask_from_request**(*request=None, args=None, kwargs=None, queue=None, **extra_options*)

app.celery.ContextTask.update_state

ContextTask.**update_state**(*task_id=None, state=None, meta=None, **kwargs*)
Update task state.

Parameters

- **task_id** (*str*) – Id of the task to update. Defaults to the id of the current task.
- **state** (*str*) – New state.
- **meta** (*Dict*) – State meta-data.

app.celery.MyCelery

```
class app.celery.MyCelery(main=None, loader=None, backend=None, amqp=None, events=None, log=None,
                           control=None, set_as_current=True, tasks=None, broker=None, include=None,
                           changes=None, config_source=None, fixups=None, task_cls=None,
                           autofinalize=True, namespace=None, strict_typing=True, **kwargs)
```

Bases: celery.app.base.Celery

Attributes

<i>MyCelery.AsyncResult</i>	Create new result instance.
<i>MyCelery.Beat</i>	celery beat scheduler application.
<i>MyCelery.GroupResult</i>	Create new group result instance.
<i>MyCelery.IS_WINDOWS</i>	
<i>MyCelery.IS_macOS</i>	
<i>MyCelery.ResultSet</i>	
<i>MyCelery.SYSTEM</i>	
<i>MyCelery.Task</i>	Base task class for this app.
<i>MyCelery.WorkController</i>	Embeddable worker.
<i>MyCelery.Worker</i>	Worker application.
<i>MyCelery.amqp</i>	@amqp.
<i>MyCelery.amqp_cls</i>	
<i>MyCelery.annotations</i>	
<i>MyCelery.backend</i>	Current backend instance.
<i>MyCelery.backend_cls</i>	
<i>MyCelery.builtin_fixups</i>	
<i>MyCelery.conf</i>	Current configuration.
<i>MyCelery.control</i>	@control.
<i>MyCelery.control_cls</i>	
<i>MyCelery.current_task</i>	Instance of task being executed, or <code>None</code> .
<i>MyCelery.current_worker_task</i>	The task currently being executed by a worker or <code>None</code> .
<i>MyCelery.events</i>	@events.
<i>MyCelery.events_cls</i>	
<i>MyCelery.loader</i>	Current loader instance.
<i>MyCelery.loader_cls</i>	
<i>MyCelery.log</i>	@log.
<i>MyCelery.log_cls</i>	
<i>MyCelery.main</i>	Name of the <code>__main__</code> module.
<i>MyCelery.oid</i>	Universally unique identifier for this app.
<i>MyCelery.on_after_configure</i>	Signal sent after app has prepared the configuration.
<i>MyCelery.on_after_finalize</i>	Signal sent after app has been finalized.
<i>MyCelery.on_after_fork</i>	Signal sent by every new process after fork.
<i>MyCelery.on_configure</i>	Signal sent when app is loading configuration.
<i>MyCelery.pool</i>	@pool.

continues on next page

Table 66 – continued from previous page

<i>MyCelery.producer_pool</i>	
<i>MyCelery.registry_cls</i>	
<i>MyCelery.steps</i>	Custom bootsteps to extend and modify the worker.
<i>MyCelery.task_cls</i>	
<i>MyCelery.tasks</i>	Task registry.
<i>MyCelery.thread_oid</i>	Per-thread unique identifier for this app.
<i>MyCelery.timezone</i>	Current timezone for this app.
<i>MyCelery.user_options</i>	Custom options for command-line programs.

app.celery.MyCelery.AsyncResult**MyCelery.AsyncResult**

Create new result instance.

See also:

`celery.result.AsyncResult`.

app.celery.MyCelery.Beat**MyCelery.Beat**

celery beat scheduler application.

See also:

`@Beat`.

app.celery.MyCelery.GroupResult**MyCelery.GroupResult**

Create new group result instance.

See also:

`celery.result.GroupResult`.

app.celery.MyCelery.IS_WINDOWS

MyCelery.IS_WINDOWS = False

app.celery.MyCelery.IS_macOS

`MyCelery.IS_macOS = False`

app.celery.MyCelery.ResultSet

`MyCelery.ResultSet`

app.celery.MyCelery.SYSTEM

`MyCelery.SYSTEM = 'Linux'`

app.celery.MyCelery.Task

`MyCelery.Task`

Base task class for this app.

app.celery.MyCelery.WorkController

`MyCelery.WorkController`

Embeddable worker.

See also:

`@WorkController`.

app.celery.MyCelery.Worker

`MyCelery.Worker`

Worker application.

See also:

`@Worker`.

app.celery.MyCelery.amqp

`MyCelery.amqp`

`@amqp`.

Type AMQP related functionality

app.celery.MyCelery.amqp_cls

```
MyCelery.amqp_cls = 'celery.app.amqp:AMQP'
```

app.celery.MyCelery.annotations

```
MyCelery.annotations
```

app.celery.MyCelery.backend

```
property MyCelery.backend
    Current backend instance.
```

app.celery.MyCelery.backend_cls

```
MyCelery.backend_cls = None
```

app.celery.MyCelery.builtin_fixups

```
MyCelery.builtin_fixups = {'celery.fixups.django:fixup'}
```

app.celery.MyCelery.conf

```
property MyCelery.conf
    Current configuration.
```

app.celery.MyCelery.control

```
MyCelery.control
    @control.
    Type Remote control
```

app.celery.MyCelery.control_cls

```
MyCelery.control_cls = 'celery.app.control:Control'
```

app.celery.MyCelery.current_task

```
property MyCelery.current_task
    Instance of task being executed, or None.
```

app.celery.MyCelery.current_worker_task**property** MyCelery.current_worker_task

The task currently being executed by a worker or None.

Differs from `current_task` in that it's not affected by tasks calling other tasks directly, or eagerly.

app.celery.MyCelery.events**MyCelery.events**

@events.

Type Consuming and sending events

app.celery.MyCelery.events_cls

MyCelery.events_cls = 'celery.app.events:Events'

app.celery.MyCelery.loader**MyCelery.loader**

Current loader instance.

app.celery.MyCelery.loader_cls

MyCelery.loader_cls = None

app.celery.MyCelery.log**MyCelery.log**

@log.

Type Logging

app.celery.MyCelery.log_cls

MyCelery.log_cls = 'celery.app.log:Logging'

app.celery.MyCelery.main**MyCelery.main = None**

Name of the `__main__` module. Required for standalone scripts.

If set this will be used instead of `__main__` when automatically generating task names.

app.celery.MyCelery.oid**MyCelery.oid**

Universally unique identifier for this app.

app.celery.MyCelery.on_after_configure**MyCelery.on_after_configure = None**

Signal sent after app has prepared the configuration.

app.celery.MyCelery.on_after_finalize**MyCelery.on_after_finalize = None**

Signal sent after app has been finalized.

app.celery.MyCelery.on_after_fork**MyCelery.on_after_fork = None**

Signal sent by every new process after fork.

app.celery.MyCelery.on_configure**MyCelery.on_configure = None**

Signal sent when app is loading configuration.

app.celery.MyCelery.pool**property MyCelery.pool**

@pool.

Note: This attribute is not related to the workers concurrency pool.

Type Broker connection pool**app.celery.MyCelery.producer_pool****property MyCelery.producer_pool**

app.celery.MyCelery.registry_cls

`MyCelery.registry_cls = 'celery.app.registry:TaskRegistry'`

app.celery.MyCelery.steps

`MyCelery.steps = None`

Custom bootsteps to extend and modify the worker. See [extending-bootsteps](#).

app.celery.MyCelery.task_cls

`MyCelery.task_cls = 'celery.app.task:Task'`

app.celery.MyCelery.tasks

`MyCelery.tasks`

Task registry.

Warning: Accessing this attribute will also auto-finalize the app.

app.celery.MyCelery.thread_oid

property `MyCelery.thread_oid`

Per-thread unique identifier for this app.

app.celery.MyCelery.timezone

`MyCelery.timezone`

Current timezone for this app.

This is a cached property taking the time zone from the `:setting:`timezone`` setting.

app.celery.MyCelery.user_options

`MyCelery.user_options = None`

Custom options for command-line programs. See [extending-commandoptions](#)

Methods

<code>MyCelery.__init__([main, loader, backend, ...])</code>	
<code>MyCelery.add_defaults(fun)</code>	Add default configuration from dict d.
<code>MyCelery.add_periodic_task(schedule, sig[, ...])</code>	
<code>MyCelery.autodiscover_tasks([packages, ...])</code>	Auto-discover task modules.
<code>MyCelery.broker_connection([hostname, ...])</code>	Establish a connection to the message broker.
<code>MyCelery.bugreport()</code>	Return information useful in bug reports.
<code>MyCelery.close()</code>	Clean up after the application.
<code>MyCelery.config_from_cmdline(argv[, namespace])</code>	
<code>MyCelery.config_from_envvar(variable_name[, ...])</code>	Read configuration from environment variable.
<code>MyCelery.config_from_object(obj[, silent, ...])</code>	Read configuration from object.
<code>MyCelery.connection([hostname, userid, ...])</code>	Establish a connection to the message broker.
<code>MyCelery.connection_for_read([url])</code>	Establish connection used for consuming.
<code>MyCelery.connection_for_write([url])</code>	Establish connection used for producing.
<code>MyCelery.connection_or_acquire([connection, ...])</code>	Context used to acquire a connection from the pool.
<code>MyCelery.create_task_cls()</code>	Create a base task class bound to this app.
<code>MyCelery.default_connection([connection, pool])</code>	Context used to acquire a connection from the pool.
<code>MyCelery.default_producer([producer])</code>	Context used to acquire a producer from the pool.
<code>MyCelery.either(default_key, *defaults)</code>	Get key from configuration or use default values.
<code>MyCelery.finalize([auto])</code>	Finalize the app.
<code>MyCelery.gen_task_name(name, module)</code>	New task default automatic naming.
<code>MyCelery.now()</code>	Return the current time and date as a datetime.
<code>MyCelery.on_init()</code>	Optional callback called at init.
<code>MyCelery.prepare_config(c)</code>	Prepare configuration before it is merged with the defaults.
<code>MyCelery.producer_or_acquire([producer])</code>	Context used to acquire a producer from the pool.
<code>MyCelery.register_task(task, **options)</code>	Utility for registering a task-based class.
<code>MyCelery.select_queues([queues])</code>	Select subset of queues.
<code>MyCelery.send_task(name[, args, kwargs, ...])</code>	Send task by name.
<code>MyCelery.set_current()</code>	Make this the current app for this thread.
<code>MyCelery.set_default()</code>	Make this the default app for all threads.
<code>MyCelery.setup_security([...])</code>	Setup the message-signing serializer.
<code>MyCelery.signature(*args, **kwargs)</code>	Return a new Signature bound to this app.
<code>MyCelery.start([argv])</code>	Run celery using <i>argv</i> .
<code>MyCelery.subclass_with_self(Class[, name, ...])</code>	Subclass an app-compatible class.

continues on next page

Table 67 – continued from previous page

<code>MyCelery.task(*args, **opts)</code>	Decorator to create a task class out of any callable.
<code>MyCelery.uses_utc_timezone()</code>	Check if the application uses the UTC time-zone.
<code>MyCelery.worker_main([argv])</code>	Run celery worker using <i>argv</i> .

app.celery.MyCelery.__init__

`MyCelery.__init__(main=None, loader=None, backend=None, amqp=None, events=None, log=None, control=None, set_as_current=True, tasks=None, broker=None, include=None, changes=None, config_source=None, fixups=None, task_cls=None, autofinalize=True, namespace=None, strict_typing=True, **kwargs)`

app.celery.MyCelery.add_defaults

`MyCelery.add_defaults(fun)`

Add default configuration from dict *d*.

If the argument is a callable function then it will be regarded as a promise, and it won't be loaded until the configuration is actually needed.

This method can be compared to:

```
>>> celery.conf.update(d)
```

with a difference that 1) no copy will be made and 2) the dict will not be transferred when the worker spawns child processes, so it's important that the same configuration happens at import time when pickle restores the object on the other side.

app.celery.MyCelery.add_periodic_task

`MyCelery.add_periodic_task(schedule, sig, args=(), kwargs=(), name=None, **opts)`

app.celery.MyCelery.autodiscover_tasks

`MyCelery.autodiscover_tasks(packages=None, related_name='tasks', force=False)`

Auto-discover task modules.

Searches a list of packages for a “tasks.py” module (or use *related_name* argument).

If the name is empty, this will be delegated to fix-ups (e.g., Django).

For example if you have a directory layout like this:

```
foo/__init__.py
   tasks.py
   models.py

bar/__init__.py
   tasks.py
```

(continues on next page)

(continued from previous page)

```
models.py

baz/__init__.py
models.py
```

Then calling `app.autodiscover_tasks(['foo', 'bar', 'baz'])` will result in the modules `foo.tasks` and `bar.tasks` being imported.

Parameters

- **packages** (*List[str]*) – List of packages to search. This argument may also be a callable, in which case the value returned is used (for lazy evaluation).
- **related_name** (*Optional[str]*) – The name of the module to find. Defaults to “tasks”: meaning “look for ‘module.tasks’ for every module in packages.”. If None will only try to import the package, i.e. “look for ‘module’”.
- **force** (*bool*) – By default this call is lazy so that the actual auto-discovery won’t happen until an application imports the default modules. Forcing will cause the auto-discovery to happen immediately.

`app.celery.MyCelery.broker_connection`

```
MyCelery.broker_connection(hostname=None, userid=None, password=None,
                           virtual_host=None, port=None, ssl=None,
                           connect_timeout=None, transport=None,
                           transport_options=None, heartbeat=None,
                           login_method=None, failover_strategy=None, **kwargs)
```

Establish a connection to the message broker.

Please use `connection_for_read()` and `connection_for_write()` instead, to convey the intent of use for this connection.

Parameters

- **url** – Either the URL or the hostname of the broker to use.
- **hostname** (*str*) – URL, Hostname/IP-address of the broker. If a URL is used, then the other argument below will be taken from the URL instead.
- **userid** (*str*) – Username to authenticate as.
- **password** (*str*) – Password to authenticate with
- **virtual_host** (*str*) – Virtual host to use (domain).
- **port** (*int*) – Port to connect to.
- **ssl** (*bool, Dict*) – Defaults to the `:setting:`broker_use_ssl`` setting.
- **transport** (*str*) – defaults to the `:setting:`broker_transport`` setting.
- **transport_options** (*Dict*) – Dictionary of transport specific options.
- **heartbeat** (*int*) – AMQP Heartbeat in seconds (pyamqp only).
- **login_method** (*str*) – Custom login method to use (AMQP only).
- **failover_strategy** (*str, Callable*) – Custom failover strategy.
- ****kwargs** – Additional arguments to `kombu.Connection`.

Returns the lazy connection instance.

Return type `kombu.Connection`

app.celery.MyCelery.bugreport**MyCelery.bugreport()**

Return information useful in bug reports.

app.celery.MyCelery.close**MyCelery.close()**

Clean up after the application.

Only necessary for dynamically created apps, and you should probably use the `with` statement instead.

Example

```
>>> with Celery(set_as_current=False) as app:
...     with app.connection_for_write() as conn:
...         pass
```

app.celery.MyCelery.config_from_cmdline**MyCelery.config_from_cmdline**(argv, namespace='celery')**app.celery.MyCelery.config_from_envvar****MyCelery.config_from_envvar**(variable_name, silent=False, force=False)

Read configuration from environment variable.

The value of the environment variable must be the name of a module to import.

Example

```
>>> os.environ['CELERY_CONFIG_MODULE'] = 'myapp.celeryconfig'
>>> celery.config_from_envvar('CELERY_CONFIG_MODULE')
```

app.celery.MyCelery.config_from_object**MyCelery.config_from_object**(obj, silent=False, force=False, namespace=None)

Read configuration from object.

Object is either an actual object or the name of a module to import.

Example

```
>>> celery.config_from_object('myapp.celeryconfig')
```

```
>>> from myapp import celeryconfig
>>> celery.config_from_object(celeryconfig)
```

Parameters

- **silent** (*bool*) – If true then import errors will be ignored.
- **force** (*bool*) – Force reading configuration immediately. By default the configuration will be read only when required.

app.celery.MyCelery.connection

`MyCelery.connection(hostname=None, userid=None, password=None, virtual_host=None, port=None, ssl=None, connect_timeout=None, transport=None, transport_options=None, heartbeat=None, login_method=None, failover_strategy=None, **kwargs)`

Establish a connection to the message broker.

Please use `connection_for_read()` and `connection_for_write()` instead, to convey the intent of use for this connection.

Parameters

- **url** – Either the URL or the hostname of the broker to use.
- **hostname** (*str*) – URL, Hostname/IP-address of the broker. If a URL is used, then the other argument below will be taken from the URL instead.
- **userid** (*str*) – Username to authenticate as.
- **password** (*str*) – Password to authenticate with
- **virtual_host** (*str*) – Virtual host to use (domain).
- **port** (*int*) – Port to connect to.
- **ssl** (*bool*, *Dict*) – Defaults to the `:setting:`broker_use_ssl`` setting.
- **transport** (*str*) – defaults to the `:setting:`broker_transport`` setting.
- **transport_options** (*Dict*) – Dictionary of transport specific options.
- **heartbeat** (*int*) – AMQP Heartbeat in seconds (pyamqp only).
- **login_method** (*str*) – Custom login method to use (AMQP only).
- **failover_strategy** (*str*, *Callable*) – Custom failover strategy.
- ****kwargs** – Additional arguments to `kombu.Connection`.

Returns the lazy connection instance.

Return type `kombu.Connection`

app.celery.MyCelery.connection_for_read

`MyCelery.connection_for_read(url=None, **kwargs)`

Establish connection used for consuming.

See also:

`connection()` for supported arguments.

app.celery.MyCelery.connection_for_write

`MyCelery.connection_for_write(url=None, **kwargs)`

Establish connection used for producing.

See also:

`connection()` for supported arguments.

app.celery.MyCelery.connection_or_acquire

`MyCelery.connection_or_acquire(connection=None, pool=True, *_ , ** __)`

Context used to acquire a connection from the pool.

For use within a `with` statement to get a connection from the pool if one is not already provided.

Parameters **connection** (*kombu.Connection*) – If not provided, a connection will be acquired from the connection pool.

app.celery.MyCelery.create_task_cls

`MyCelery.create_task_cls()`

Create a base task class bound to this app.

app.celery.MyCelery.default_connection

`MyCelery.default_connection(connection=None, pool=True, *_ , ** __)`

Context used to acquire a connection from the pool.

For use within a `with` statement to get a connection from the pool if one is not already provided.

Parameters **connection** (*kombu.Connection*) – If not provided, a connection will be acquired from the connection pool.

app.celery.MyCelery.default_producer

`MyCelery.default_producer(producer=None)`

Context used to acquire a producer from the pool.

For use within a `with` statement to get a producer from the pool if one is not already provided

Parameters **producer** (*kombu.Producer*) – If not provided, a producer will be acquired from the producer pool.

app.celery.MyCelery.either

`MyCelery.either(default_key, *defaults)`

Get key from configuration or use default values.

Fallback to the value of a configuration key if none of the **values* are true.

app.celery.MyCelery.finalize**MyCelery.finalize**(*auto=False*)

Finalize the app.

This loads built-in tasks, evaluates pending task decorators, reads configuration, etc.

app.celery.MyCelery.gen_task_name**MyCelery.gen_task_name**(*name, module*)

New task default automatic naming.

The default `gen_task_name` method builds task names based on absolute imports, for example:**project** / `/__init__.py` /`moduleA/`
`/__init.py` /`tasks.py``/moduleB/` `/__init.py` /`tasks.py`The default automatic naming is “`project.moduleA.tasks.taskA`”, “`project.moduleA.tasks.taskB`”, etc. This new default automatic naming forget “tasks” in all task names:

DEFAULT	WAY	NEW	WAY	project.moduleA.tasks.taskA	project.moduleA.taskA
project.moduleA.tasks.taskA			project.moduleA.taskB		project.moduleB.tasks.taskA
project.moduleB.taskA					

This method is only used when the tasks don’t have a name attribute defined, otherwise, the task name will be respect.

Referenceshttps://docs.celeryproject.org/en/stable/userguide/tasks.html?highlight=gen_task_name#changing-the-automatic-naming-behavior**app.celery.MyCelery.now****MyCelery.now**()

Return the current time and date as a datetime.

app.celery.MyCelery.on_init**MyCelery.on_init**()

Optional callback called at init.

app.celery.MyCelery.prepare_config**MyCelery.prepare_config(c)**

Prepare configuration before it is merged with the defaults.

app.celery.MyCelery.producer_or_acquire**MyCelery.producer_or_acquire(producer=None)**

Context used to acquire a producer from the pool.

For use within a `with` statement to get a producer from the pool if one is not already provided**Parameters** **producer** (*kombu.Producer*) – If not provided, a producer will be acquired from the producer pool.**app.celery.MyCelery.register_task****MyCelery.register_task(task, **options)**

Utility for registering a task-based class.

Note: This is here for compatibility with old Celery 1.0 style task classes, you should not need to use this for new projects.

app.celery.MyCelery.select_queues**MyCelery.select_queues(queues=None)**

Select subset of queues.

Parameters **queues** (*Sequence[str]*) – a list of queue names to keep.**app.celery.MyCelery.send_task****MyCelery.send_task(name, args=None, kwargs=None, countdown=None, eta=None, task_id=None, producer=None, connection=None, router=None, result_cls=None, expires=None, publisher=None, link=None, link_error=None, add_to_parent=True, group_id=None, group_index=None, retries=0, chord=None, reply_to=None, time_limit=None, soft_time_limit=None, root_id=None, parent_id=None, route_name=None, shadow=None, chain=None, task_type=None, **options)**

Send task by name.

Supports the same arguments as `@-Task.apply_async()`.**Parameters**

- **name** (*str*) – Name of task to call (e.g., “*tasks.add*”).
- **result_cls** (*AsyncResult*) – Specify custom result class.

app.celery.MyCelery.set_current

MyCelery.set_current()

Make this the current app for this thread.

app.celery.MyCelery.set_default

MyCelery.set_default()

Make this the default app for all threads.

app.celery.MyCelery.setup_security

MyCelery.setup_security(*allowed_serializers=None, key=None, cert=None, store=None, digest='sha256', serializer='json'*)

Setup the message-signing serializer.

This will affect all application instances (a global operation).

Disables untrusted serializers and if configured to use the auth serializer will register the auth serializer with the provided settings into the Kombu serializer registry.

Parameters

- **allowed_serializers** (*Set[str]*) – List of serializer names, or content_types that should be exempt from being disabled.
- **key** (*str*) – Name of private key file to use. Defaults to the **:setting: security_key** setting.
- **cert** (*str*) – Name of certificate file to use. Defaults to the **:setting: security_certificate** setting.
- **store** (*str*) – Directory containing certificates. Defaults to the **:setting: security_cert_store** setting.
- **digest** (*str*) – Digest algorithm used when signing messages. Default is sha256.
- **serializer** (*str*) – Serializer used to encode messages after they've been signed. See **:setting: task_serializer** for the serializers supported. Default is json.

app.celery.MyCelery.signature

MyCelery.signature(*args, **kwargs)

Return a new Signature bound to this app.

app.celery.MyCelery.start

MyCelery.start(argv=None)

Run **celery** using *argv*.

Uses `sys.argv` if *argv* is not specified.

app.celery.MyCelery.subclass_with_self

`MyCelery.subclass_with_self(Class, name=None, attribute='app', reverse=None, keep_reduce=False, **kw)`

Subclass an app-compatible class.

App-compatible means that the class has a class attribute that provides the default app it should use, for example: `class Foo: app = None`.

Parameters

- **Class** (*type*) – The app-compatible class to subclass.
- **name** (*str*) – Custom name for the target class.
- **attribute** (*str*) – Name of the attribute holding the app, Default is 'app'.
- **reverse** (*str*) – Reverse path to this object used for pickling purposes. For example, to get `app.AsyncResult`, use `"AsyncResult"`.
- **keep_reduce** (*bool*) – If enabled a custom `__reduce__` implementation won't be provided.

app.celery.MyCelery.task

`MyCelery.task(*args, **opts)`

Decorator to create a task class out of any callable.

See Task options for a list of the arguments that can be passed to this decorator.

Examples

```
@app.task
def refresh_feed(url):
    store_feed(feedparser.parse(url))
```

with setting extra options:

```
@app.task(exchange='feeds')
def refresh_feed(url):
    return store_feed(feedparser.parse(url))
```

Note: App Binding: For custom apps the task decorator will return a proxy object, so that the act of creating the task is not performed until the task is used or the task registry is accessed.

If you're depending on binding to be deferred, then you must not access any attributes on the returned object until the application is fully set up (finalized).

app.celery.MyCelery.uses_utc_timezone**MyCelery.uses_utc_timezone()**

Check if the application uses the UTC timezone.

app.celery.MyCelery.worker_main**MyCelery.worker_main**(*argv=None*)Run **celery worker** using *argv*.Uses `sys.argv` if *argv* is not specified.**Functions**

make_celery(app)

app.celery.make_celery**app.celery.make_celery**(*app: flask.app.Flask*) → `celery.app.base.Celery`**class app.celery.ContextTask****AsyncResult**(*task_id, **kwargs*)

Get AsyncResult instance for the specified task.

Parameters **task_id** (*str*) – Task id to get result for.**exception MaxRetriesExceededError**(**args, **kwargs*)

The tasks max restart limit has been exceeded.

args**with_traceback**()

Exception.with_traceback(tb) – set self.__traceback__ to tb and return self.

exception OperationalError

Recoverable message transport connection error.

args**with_traceback**()

Exception.with_traceback(tb) – set self.__traceback__ to tb and return self.

Request = **'celery.worker.request:Request'**

Request class used, or the qualified name of one.

Strategy = **'celery.worker.strategy:default'**

Execution strategy used, or the qualified name of one.

_app = **<MyCelery __main__>**

The application instance associated with this task class.

_backend = **None**

_default_request = None

Some may expect a request to exist even if the task hasn't been called. This should probably be deprecated.

_exec_options = None

classmethod _get_app()

_get_exec_options()

_get_request()

Get current request object.

abstract = True

Deprecated attribute abstract here for compatibility.

acks_late = False

When enabled messages for this task will be acknowledged **after** the task has been executed, and not *just before* (the default behavior).

Please note that this means the task may be executed twice if the worker crashes mid execution.

The application default can be overridden with the `:setting:`task_acks_late`` setting.

acks_on_failure_or_timeout = True

When enabled messages for this task will be acknowledged even if it fails or times out.

Configuring this setting only applies to tasks that are acknowledged **after** they have been executed and only if `:setting:`task_acks_late`` is enabled.

The application default can be overridden with the `:setting:`task_acks_on_failure_or_timeout`` setting.

classmethod add_around(attr, around)

add_to_chord(sig, lazy=False)

Add signature to the chord the current task is a member of.

New in version 4.0.

Currently only supported by the Redis result backend.

Parameters

- **sig** (*Signature*) – Signature to extend chord with.
- **lazy** (*bool*) – If enabled the new task won't actually be called, and `sig.delay()` must be called manually.

add_trail(result)

after_return(status, retval, task_id, args, kwargs, einfo)

Handler called after the task returns.

Parameters

- **status** (*str*) – Current task state.
- **retval** (*Any*) – Task return value/exception.
- **task_id** (*str*) – Unique id of the task.
- **args** (*Tuple*) – Original arguments for the task.
- **kwargs** (*Dict*) – Original keyword arguments for the task.
- **einfo** (*ExceptionInfo*) – Exception information.

Returns The return value of this handler is ignored.

Return type None

classmethod `annotate()`

`app = <MyCelery __main__>`

apply(*args=None, kwargs=None, link=None, link_error=None, task_id=None, retries=None, throw=None, logfile=None, loglevel=None, headers=None, **options*)

Execute this task locally, by blocking until the task returns.

Parameters

- **args** (*Tuple*) – positional arguments passed on to the task.
- **kwargs** (*Dict*) – keyword arguments passed on to the task.
- **throw** (*bool*) – Re-raise task exceptions. Defaults to the `:setting:`task_eager_propagates`` setting.

Returns pre-evaluated result.

Return type `celery.result.EagerResult`

apply_async(*args=None, kwargs=None, task_id=None, producer=None, link=None, link_error=None, shadow=None, **options*)

Apply tasks asynchronously by sending a message.

Parameters

- **args** (*Tuple*) – The positional arguments to pass on to the task.
- **kwargs** (*Dict*) – The keyword arguments to pass on to the task.
- **countdown** (*float*) – Number of seconds into the future that the task should execute. Defaults to immediate execution.
- **eta** (*datetime*) – Absolute time and date of when the task should be executed. May not be specified if *countdown* is also supplied.
- **expires** (*float, datetime*) – Datetime or seconds in the future for the task should expire. The task won't be executed after the expiration time.
- **shadow** (*str*) – Override task name used in logs/monitoring. Default is retrieved from `shadow_name()`.
- **connection** (*kombu.Connection*) – Re-use existing broker connection instead of acquiring one from the connection pool.
- **retry** (*bool*) – If enabled sending of the task message will be retried in the event of connection loss or failure. Default is taken from the `:setting:`task_publish_retry`` setting. Note that you need to handle the producer/connection manually for this to work.
- **retry_policy** (*Mapping*) – Override the retry policy used. See the `:setting:`task_publish_retry_policy`` setting.
- **time_limit** (*int*) – If set, overrides the default time limit.
- **soft_time_limit** (*int*) – If set, overrides the default soft time limit.
- **queue** (*str, kombu.Queue*) – The queue to route the task to. This must be a key present in `:setting:`task_queues``, or `:setting:`task_create_missing_queues`` must be enabled. See guide-routing for more information.
- **exchange** (*str, kombu.Exchange*) – Named custom exchange to send the task to. Usually not used in combination with the *queue* argument.

- **routing_key** (*str*) – Custom routing key used to route the task to a worker server. If in combination with a `queue` argument only used to specify custom routing keys to topic exchanges.
- **priority** (*int*) – The task priority, a number between 0 and 9. Defaults to the `priority` attribute.
- **serializer** (*str*) – Serialization method to use. Can be `pickle`, `json`, `yaml`, `msgpack` or any custom serialization method that's been registered with `kombu.serialization.registry`. Defaults to the `serializer` attribute.
- **compression** (*str*) – Optional compression method to use. Can be one of `zlib`, `bzip2`, or any custom compression methods registered with `kombu.compression.register()`. Defaults to the `:setting:`task_compression`` setting.
- **link** (*Signature*) – A single, or a list of tasks signatures to apply if the task returns successfully.
- **link_error** (*Signature*) – A single, or a list of task signatures to apply if an error occurs while executing the task.
- **producer** (*kombu.Producer*) – custom producer to use when publishing the task.
- **add_to_parent** (*bool*) – If set to `True` (default) and the task is applied while executing another task, then the result will be appended to the parent tasks `request.children` attribute. Trailing can also be disabled by default using the `trail` attribute
- **ignore_result** (*bool*) – If set to `False` (default) the result of a task will be stored in the backend. If set to `True` the result will not be stored. This can also be set using the `ignore_result` in the `app.task` decorator.
- **publisher** (*kombu.Producer*) – Deprecated alias to `producer`.
- **headers** (*Dict*) – Message headers to be included in the message.

Returns Promise of future evaluation.

Return type `celery.result.AsyncResult`

Raises

- **TypeError** – If not enough arguments are passed, or too many arguments are passed. Note that signature checks may be disabled by specifying `@task(typing=False)`.
- **`kombu.exceptions.OperationalError`** – If a connection to the transport cannot be made, or if the connection is lost.

Note: Also supports all keyword arguments supported by `kombu.Producer.publish()`.

property backend

before_start(*task_id*, *args*, *kwargs*)

Handler called before the task starts.

New in version 5.2.

Parameters

- **task_id** (*str*) – Unique id of the task to execute.
- **args** (*Tuple*) – Original arguments for the task to execute.
- **kwargs** (*Dict*) – Original keyword arguments for the task to execute.

Returns The return value of this handler is ignored.

Return type None

classmethod `bind(app)`

chunks(*it*, *n*)

Create a chunks task for this task.

default_retry_delay = 180

Default time in seconds before a retry of the task should be executed. 3 minutes by default.

delay(*args, **kwargs)

Star argument version of `apply_async()`.

Does not support the extra options enabled by `apply_async()`.

Parameters

- ***args** (*Any*) – Positional arguments passed on to the task.
- ****kwargs** (*Any*) – Keyword arguments passed on to the task.

Returns Future promise.

Return type `celery.result.AsyncResult`

expires = None

Default task expiry time.

from_config = (('serializer', 'task_serializer'), ('rate_limit', 'task_default_rate_limit'), ('priority', 'task_default_priority'), ('track_started', 'task_track_started'), ('acks_late', 'task_acks_late'), ('acks_on_failure_or_timeout', 'task_acks_on_failure_or_timeout'), ('reject_on_worker_lost', 'task_reject_on_worker_lost'), ('ignore_result', 'task_ignore_result'), ('store_eager_result', 'task_store_eager_result'), ('store_errors_even_if_ignored', 'task_store_errors_even_if_ignored'))

ignore_result = False

If enabled the worker won't store task state and return values for this task. Defaults to the **setting: `task_ignore_result`** setting.

map(*it*)

Create a xmap task from it.

max_retries = 3

Maximum number of retries before giving up. If set to None, it will **never** stop retrying.

name = None

Name of the task.

classmethod `on_bound(app)`

Called when the task is bound to an app.

Note: This class method can be defined to do additional actions when the task class is bound to an app.

on_failure(*exc*, *task_id*, *args*, *kwargs*, *info*) → None

Error handler.

This is run by the worker when the task fails.

Parameters

- **exc** (*Exception*) – The exception raised by the task.
- **task_id** (*str*) – Unique id of the failed task.
- **args** (*Tuple*) – Original arguments for the task that failed.
- **kwargs** (*Dict*) – Original keyword arguments for the task that failed.
- **einfo** (*ExceptionInfo*) – Exception information.

Returns The return value of this handler is ignored.

Return type None

on_retry(*exc, task_id, args, kwargs, einfo*)

Retry handler.

This is run by the worker when the task is to be retried.

Parameters

- **exc** (*Exception*) – The exception sent to `retry()`.
- **task_id** (*str*) – Unique id of the retried task.
- **args** (*Tuple*) – Original arguments for the retried task.
- **kwargs** (*Dict*) – Original keyword arguments for the retried task.
- **einfo** (*ExceptionInfo*) – Exception information.

Returns The return value of this handler is ignored.

Return type None

on_success(*retval, task_id, args, kwargs*)

Success handler.

Run by the worker if the task executes successfully.

Parameters

- **retval** (*Any*) – The return value of the task.
- **task_id** (*str*) – Unique id of the executed task.
- **args** (*Tuple*) – Original arguments for the executed task.
- **kwargs** (*Dict*) – Original keyword arguments for the executed task.

Returns The return value of this handler is ignored.

Return type None

pop_request()

priority = None

Default task priority.

push_request(**args*, ***kwargs*)

rate_limit = None

None (no rate limit), '100/s' (hundred tasks a second), '100/m' (hundred tasks a minute), '100/h' (hundred tasks an hour)

Type Rate limit for this task type. Examples

reject_on_worker_lost = None

Even if `acks_late` is enabled, the worker will acknowledge tasks when the worker process executing them abruptly exits or is signaled (e.g., `:sig:KILL`/:sig:INT``, etc).

Setting this to true allows the message to be re-queued instead, so that the task will execute again by the same worker, or another worker.

Warning: Enabling this can cause message loops; make sure you know what you're doing.

replace(*sig*)

Replace this task, with a new task inheriting the task id.

Execution of the host task ends immediately and no subsequent statements will be run.

New in version 4.0.

Parameters *sig* (*Signature*) – signature to replace with.

Raises

- `~@Ignore` – This is always raised when called in asynchronous context.
- It is best to always use `return self.replace(...)` to convey –
- to the reader that the task won't continue after being replaced. –

property request

Get current request object.

request_stack = <celery.utils.threads._LocalStack object>

Task request stack, the current request will be the topmost.

resultrepr_maxsize = 1024

Max length of result representation used in logs and events.

retry(*args=None, kwargs=None, exc=None, throw=True, eta=None, countdown=None, max_retries=None, **options*)

Retry the task, adding it to the back of the queue.

Example

```
>>> from imaginary_twitter_lib import Twitter
>>> from proj.celery import app
```

```
>>> @app.task(bind=True)
... def tweet(self, auth, message):
...     twitter = Twitter(oauth=auth)
...     try:
...         twitter.post_status_update(message)
...     except twitter.FailWhale as exc:
...         # Retry in 5 minutes.
...         self.retry(countdown=60 * 5, exc=exc)
```

Note: Although the task will never return above as `retry` raises an exception to notify the worker, we use `raise` in front of the `retry` to convey that the rest of the block won't be executed.

Parameters

- **args** (*Tuple*) – Positional arguments to retry with.
- **kwargs** (*Dict*) – Keyword arguments to retry with.
- **exc** (*Exception*) – Custom exception to report when the max retry limit has been exceeded (default: `@MaxRetriesExceededError`).

If this argument is set and retry is called while an exception was raised (`sys.exc_info()` is set) it will attempt to re-raise the current exception.

If no exception was raised it will raise the **exc** argument provided.

- **countdown** (*float*) – Time in seconds to delay the retry for.
- **eta** (*datetime*) – Explicit time and date to run the retry at.
- **max_retries** (*int*) – If set, overrides the default retry limit for this execution. Changes to this parameter don't propagate to subsequent task retry attempts. A value of `None`, means “use the default”, so if you want infinite retries you'd have to set the `max_retries` attribute of the task to `None` first.
- **time_limit** (*int*) – If set, overrides the default time limit.
- **soft_time_limit** (*int*) – If set, overrides the default soft time limit.
- **throw** (*bool*) – If this is `False`, don't raise the `@Retry` exception, that tells the worker to mark the task as being retried. Note that this means the task will be marked as failed if the task raises an exception, or successful if it returns after the retry call.
- ****options** (*Any*) – Extra options to pass on to `apply_async()`.

Raises `celery.exceptions.Retry` – To tell the worker that the task has been re-sent for retry. This always happens, unless the `throw` keyword argument has been explicitly set to `False`, and is considered normal operation.

run (**args, **kwargs*)

The body of the task executed by workers.

s (**args, **kwargs*)

Create signature.

Shortcut for `.s(*a, **k) -> .signature(a, k)`.

send_event (*type_, retry=True, retry_policy=None, **fields*)

Send monitoring event message.

This can be used to add custom event types in `:pypi:Flower`` and other monitors.

Parameters **type** (*str*) – Type of event, e.g. "task-failed".

Keyword Arguments

- **retry** (*bool*) – Retry sending the message if the connection is lost. Default is taken from the `:setting:task_publish_retry`` setting.
- **retry_policy** (*Mapping*) – Retry settings. Default is taken from the `:setting:task_publish_retry_policy`` setting.
- ****fields** (*Any*) – Map containing information about the event. Must be JSON serializable.

send_events = True

If enabled the worker will send monitoring events related to this task (but only if the worker is configured to send task related events). Note that this has no effect on the task-failure event case where a task is not registered (as it will have no task class to check this flag).

serializer = 'json'

The name of a serializer that are registered with `kombu.serialization.registry`. Default is `'json'`.

shadow_name(args, kwargs, options)

Override for custom task name in worker logs/monitoring.

Example

```
from celery.utils.imports import qualname

def shadow_name(task, args, kwargs, options):
    return qualname(args[0])

@app.task(shadow_name=shadow_name, serializer='pickle')
def apply_function_async(fun, *args, **kwargs):
    return fun(*args, **kwargs)
```

Parameters

- **args** (*Tuple*) – Task positional arguments.
- **kwargs** (*Dict*) – Task keyword arguments.
- **options** (*Dict*) – Task execution options.

si(*args, **kwargs)

Create immutable signature.

Shortcut for `.si(*a, **k) -> .signature(a, k, immutable=True)`.

signature(args=None, *starargs, **starkwargs)

Create signature.

Returns

object for this task, wrapping arguments and execution options for a single task invocation.

Return type signature

signature_from_request(request=None, args=None, kwargs=None, queue=None, **extra_options)

soft_time_limit = None

Soft time limit. Defaults to the **setting: `task_soft_time_limit`** setting.

starmap(it)

Create a xstarmap task from it.

start_strategy(app, consumer, **kwargs)

store_eager_result = False

store_errors_even_if_ignored = False

When enabled errors will be stored even if the task is otherwise configured to ignore results.

subtask(args=None, *starargs, **starkwargs)

Create signature.

Returns

object for this task, wrapping arguments and execution options for a single task invocation.

Return type signature

subtask_from_request(*request=None, args=None, kwargs=None, queue=None, **extra_options*)

throws = ()

Tuple of expected exceptions.

These are errors that are expected in normal operation and that shouldn't be regarded as a real error by the worker. Currently this means that the state will be updated to an error state, but the worker won't log the event as an error.

time_limit = None

Hard time limit. Defaults to the **:setting:`task_time_limit`** setting.

track_started = False

If enabled the task will report its status as 'started' when the task is executed by a worker. Disabled by default as the normal behavior is to not report that level of granularity. Tasks are either pending, finished, or waiting to be retried.

Having a 'started' status can be useful for when there are long running tasks and there's a need to report what task is currently running.

The application default can be overridden using the **:setting:`task_track_started`** setting.

trail = True

If enabled the request will keep track of subtasks started by this task, and this information will be sent with the result (`result.children`).

typing = True

Enable argument checking. You can set this to false if you don't want the signature to be checked when calling the task. Defaults to `app.strict_typing`.

update_state(*task_id=None, state=None, meta=None, **kwargs*)

Update task state.

Parameters

- **task_id** (*str*) – Id of the task to update. Defaults to the id of the current task.
- **state** (*str*) – New state.
- **meta** (*Dict*) – State meta-data.

```
class app.celery.MyCelery(main=None, loader=None, backend=None, amqp=None, events=None, log=None,
                           control=None, set_as_current=True, tasks=None, broker=None, include=None,
                           changes=None, config_source=None, fixups=None, task_cls=None,
                           autofinalize=True, namespace=None, strict_typing=True, **kwargs)
```

AsyncResult

Create new result instance.

See also:

`celery.result.AsyncResult`.

Beat

celery beat scheduler application.

See also:

`@Beat`.

GroupResult

Create new group result instance.

See also:

`celery.result.GroupResult`.

IS_WINDOWS = False

IS_macOS = False

Pickler

alias of `celery.app.utils.AppPickler`

ResultSet

SYSTEM = 'Linux'

Task

Base task class for this app.

WorkController

Embeddable worker.

See also:

`@WorkController`.

Worker

Worker application.

See also:

`@Worker`.

`_acquire_connection(pool=True)`

Helper for `connection_or_acquire()`.

`_add_periodic_task(key, entry)`

`_after_fork()`

`_after_fork_registered = False`

`_autodiscover_tasks(packages, related_name, **kwargs)`

`_autodiscover_tasks_from_fixups(related_name)`

`_autodiscover_tasks_from_names(packages, related_name)`

`_canvas`

`_conf = None`

`_connection(url, userid=None, password=None, virtual_host=None, port=None, ssl=None, connect_timeout=None, transport=None, transport_options=None, heartbeat=None, login_method=None, failover_strategy=None, **kwargs)`

`_ensure_after_fork()`

`_finalize_pending_conf()`

Get config value by key and finalize loading the configuration.

Note:

This is used by PendingConfiguration: as soon as you access a key the configuration is read.

`_fixups = None`

```

_get_backend()
_get_default_loader()
_load_config()
_local = None
    Thread local storage.
_pool = None
_rgetattr(path)
_sig_to_periodic_task_entry(schedule, sig, args=(), kwargs=None, name=None, **opts)
_task_from_fun(fun, name=None, base=None, bind=False, **options)

```

add_defaults(fun)

Add default configuration from dict d.

If the argument is a callable function then it will be regarded as a promise, and it won't be loaded until the configuration is actually needed.

This method can be compared to:

```
>>> celery.conf.update(d)
```

with a difference that 1) no copy will be made and 2) the dict will not be transferred when the worker spawns child processes, so it's important that the same configuration happens at import time when pickle restores the object on the other side.

```
add_periodic_task(schedule, sig, args=(), kwargs=(), name=None, **opts)
```

amqp

@amqp.

Type AMQP related functionality

```
amqp_cls = 'celery.app.amqp:AMQP'
```

annotations

```
autodiscover_tasks(packages=None, related_name='tasks', force=False)
```

Auto-discover task modules.

Searches a list of packages for a “tasks.py” module (or use related_name argument).

If the name is empty, this will be delegated to fix-ups (e.g., Django).

For example if you have a directory layout like this:

```

foo/__init__.py
    tasks.py
    models.py

bar/__init__.py
    tasks.py
    models.py

baz/__init__.py
    models.py

```

Then calling `app.autodiscover_tasks(['foo', 'bar', 'baz'])` will result in the modules `foo.tasks` and `bar.tasks` being imported.

Parameters

- **packages** (*List[str]*) – List of packages to search. This argument may also be a callable, in which case the value returned is used (for lazy evaluation).
- **related_name** (*Optional[str]*) – The name of the module to find. Defaults to “tasks”: meaning “look for ‘module.tasks’ for every module in packages.”. If None will only try to import the package, i.e. “look for ‘module’”.
- **force** (*bool*) – By default this call is lazy so that the actual auto-discovery won’t happen until an application imports the default modules. Forcing will cause the auto-discovery to happen immediately.

property backend

Current backend instance.

backend_cls = None

broker_connection(*hostname=None, userid=None, password=None, virtual_host=None, port=None, ssl=None, connect_timeout=None, transport=None, transport_options=None, heartbeat=None, login_method=None, failover_strategy=None, **kwargs*)

Establish a connection to the message broker.

Please use [`connection_for_read\(\)`](#) and [`connection_for_write\(\)`](#) instead, to convey the intent of use for this connection.

Parameters

- **url** – Either the URL or the hostname of the broker to use.
- **hostname** (*str*) – URL, Hostname/IP-address of the broker. If a URL is used, then the other argument below will be taken from the URL instead.
- **userid** (*str*) – Username to authenticate as.
- **password** (*str*) – Password to authenticate with
- **virtual_host** (*str*) – Virtual host to use (domain).
- **port** (*int*) – Port to connect to.
- **ssl** (*bool, Dict*) – Defaults to the `:setting:`broker_use_ssl`` setting.
- **transport** (*str*) – defaults to the `:setting:`broker_transport`` setting.
- **transport_options** (*Dict*) – Dictionary of transport specific options.
- **heartbeat** (*int*) – AMQP Heartbeat in seconds (pyamqp only).
- **login_method** (*str*) – Custom login method to use (AMQP only).
- **failover_strategy** (*str, Callable*) – Custom failover strategy.
- ****kwargs** – Additional arguments to `kombu.Connection`.

Returns the lazy connection instance.

Return type `kombu.Connection`

bugreport()

Return information useful in bug reports.

builtin_fixups = `{'celery.fixups.django:fixup'}`

close()

Clean up after the application.

Only necessary for dynamically created apps, and you should probably use the `with` statement instead.

Example

```
>>> with Celery(set_as_current=False) as app:
...     with app.connection_for_write() as conn:
...         pass
```

property `conf`

Current configuration.

`config_from_cmdline`(*argv*, *namespace*='celery')

`config_from_envvar`(*variable_name*, *silent*=False, *force*=False)

Read configuration from environment variable.

The value of the environment variable must be the name of a module to import.

Example

```
>>> os.environ['CELERY_CONFIG_MODULE'] = 'myapp.celeryconfig'
>>> celery.config_from_envvar('CELERY_CONFIG_MODULE')
```

`config_from_object`(*obj*, *silent*=False, *force*=False, *namespace*=None)

Read configuration from object.

Object is either an actual object or the name of a module to import.

Example

```
>>> celery.config_from_object('myapp.celeryconfig')
```

```
>>> from myapp import celeryconfig
>>> celery.config_from_object(celeryconfig)
```

Parameters

- **`silent`** (*bool*) – If true then import errors will be ignored.
- **`force`** (*bool*) – Force reading configuration immediately. By default the configuration will be read only when required.

`connection`(*hostname*=None, *userid*=None, *password*=None, *virtual_host*=None, *port*=None, *ssl*=None, *connect_timeout*=None, *transport*=None, *transport_options*=None, *heartbeat*=None, *login_method*=None, *failover_strategy*=None, ***kwargs*)

Establish a connection to the message broker.

Please use `connection_for_read()` and `connection_for_write()` instead, to convey the intent of use for this connection.

Parameters

- **`url`** – Either the URL or the hostname of the broker to use.

- **hostname** (*str*) – URL, Hostname/IP-address of the broker. If a URL is used, then the other argument below will be taken from the URL instead.
- **userid** (*str*) – Username to authenticate as.
- **password** (*str*) – Password to authenticate with
- **virtual_host** (*str*) – Virtual host to use (domain).
- **port** (*int*) – Port to connect to.
- **ssl** (*bool*, *Dict*) – Defaults to the **:setting:`broker_use_ssl`** setting.
- **transport** (*str*) – defaults to the **:setting:`broker_transport`** setting.
- **transport_options** (*Dict*) – Dictionary of transport specific options.
- **heartbeat** (*int*) – AMQP Heartbeat in seconds (pyamqp only).
- **login_method** (*str*) – Custom login method to use (AMQP only).
- **failover_strategy** (*str*, *Callable*) – Custom failover strategy.
- ****kwargs** – Additional arguments to `kombu.Connection`.

Returns the lazy connection instance.

Return type `kombu.Connection`

connection_for_read(*url=None*, ***kwargs*)

Establish connection used for consuming.

See also:

[`connection\(\)`](#) for supported arguments.

connection_for_write(*url=None*, ***kwargs*)

Establish connection used for producing.

See also:

[`connection\(\)`](#) for supported arguments.

connection_or_acquire(*connection=None*, *pool=True*, **_, **__*)

Context used to acquire a connection from the pool.

For use within a `with` statement to get a connection from the pool if one is not already provided.

Parameters **connection** (`kombu.Connection`) – If not provided, a connection will be acquired from the connection pool.

control

@control.

Type Remote control

control_cls = 'celery.app.control:Control'

create_task_cls()

Create a base task class bound to this app.

property current_task

Instance of task being executed, or None.

property current_worker_task

The task currently being executed by a worker or None.

Differs from [`current_task`](#) in that it's not affected by tasks calling other tasks directly, or eagerly.

default_connection(*connection=None, pool=True, *, **__*)

Context used to acquire a connection from the pool.

For use within a `with` statement to get a connection from the pool if one is not already provided.

Parameters **connection** (*kombu.Connection*) – If not provided, a connection will be acquired from the connection pool.

default_producer(*producer=None*)

Context used to acquire a producer from the pool.

For use within a `with` statement to get a producer from the pool if one is not already provided

Parameters **producer** (*kombu.Producer*) – If not provided, a producer will be acquired from the producer pool.

either(*default_key, *defaults*)

Get key from configuration or use default values.

Fallback to the value of a configuration key if none of the **values* are true.

events

@events.

Type Consuming and sending events

events_cls = 'celery.app.events:Events'

finalize(*auto=False*)

Finalize the app.

This loads built-in tasks, evaluates pending task decorators, reads configuration, etc.

gen_task_name(*name, module*)

New task default automatic naming.

The default `gen_task_name` method builds task names based on absolute imports, for example:

project / `/__init__.py` /`moduleA/`

`/__init.py` /`tasks.py`

/moduleB/ /`/__init.py` /`tasks.py`

The default automatic naming is “`project.moduleA.tasks.taskA`”, “`project.moduleA.tasks.taskB`”, etc. This new default automatic naming forget “`tasks`” in all task names:

DEFAULT	WAY	NEW	WAY	<code>project.moduleA.tasks.taskA</code>	<code>project.moduleA.taskA</code>
				<code>project.moduleA.tasks.taskA</code>	<code>project.moduleA.taskB</code>
				<code>project.moduleB.tasks.taskA</code>	<code>project.moduleB.taskA</code>

This method is only used when the tasks don’t have a name attribute defined, otherwise, the task name will be respect.

References

https://docs.celeryproject.org/en/stable/userguide/tasks.html?highlight=gen_task_name#changing-the-automatic-naming-behavior

loader

Current loader instance.

loader_cls = None

log

@log.

Type Logging

log_cls = 'celery.app.log:Logging'

main = None

Name of the `__main__` module. Required for standalone scripts.

If set this will be used instead of `__main__` when automatically generating task names.

now()

Return the current time and date as a datetime.

oid

Universally unique identifier for this app.

on_after_configure = None

Signal sent after app has prepared the configuration.

on_after_finalize = None

Signal sent after app has been finalized.

on_after_fork = None

Signal sent by every new process after fork.

on_configure = None

Signal sent when app is loading configuration.

on_init()

Optional callback called at init.

property pool

@pool.

Note: This attribute is not related to the workers concurrency pool.

Type Broker connection pool

prepare_config(c)

Prepare configuration before it is merged with the defaults.

producer_or_acquire(producer=None)

Context used to acquire a producer from the pool.

For use within a `with` statement to get a producer from the pool if one is not already provided

Parameters **producer** (*kombu.Producer*) – If not provided, a producer will be acquired from the producer pool.

property producer_pool

register_task(*task*, ***options*)
Utility for registering a task-based class.

Note: This is here for compatibility with old Celery 1.0 style task classes, you should not need to use this for new projects.

registry_cls = 'celery.app.registry:TaskRegistry'

select_queues(*queues=None*)
Select subset of queues.

Parameters *queues* (*Sequence[str]*) – a list of queue names to keep.

send_task(*name*, *args=None*, *kwargs=None*, *countdown=None*, *eta=None*, *task_id=None*, *producer=None*, *connection=None*, *router=None*, *result_cls=None*, *expires=None*, *publisher=None*, *link=None*, *link_error=None*, *add_to_parent=True*, *group_id=None*, *group_index=None*, *retries=0*, *chord=None*, *reply_to=None*, *time_limit=None*, *soft_time_limit=None*, *root_id=None*, *parent_id=None*, *route_name=None*, *shadow=None*, *chain=None*, *task_type=None*, ***options*)
Send task by name.

Supports the same arguments as `@-Task.apply_async()`.

Parameters

- **name** (*str*) – Name of task to call (e.g., “tasks.add”).
- **result_cls** (*AsyncResult*) – Specify custom result class.

set_current()
Make this the current app for this thread.

set_default()
Make this the default app for all threads.

setup_security(*allowed_serializers=None*, *key=None*, *cert=None*, *store=None*, *digest='sha256'*, *serializer='json'*)
Setup the message-signing serializer.

This will affect all application instances (a global operation).

Disables untrusted serializers and if configured to use the `auth` serializer will register the `auth` serializer with the provided settings into the Kombu serializer registry.

Parameters

- **allowed_serializers** (*Set[str]*) – List of serializer names, or content_types that should be exempt from being disabled.
- **key** (*str*) – Name of private key file to use. Defaults to the `:setting:`security_key`` setting.
- **cert** (*str*) – Name of certificate file to use. Defaults to the `:setting:`security_certificate`` setting.
- **store** (*str*) – Directory containing certificates. Defaults to the `:setting:`security_cert_store`` setting.
- **digest** (*str*) – Digest algorithm used when signing messages. Default is sha256.
- **serializer** (*str*) – Serializer used to encode messages after they’ve been signed. See `:setting:`task_serializer`` for the serializers supported. Default is json.

signature(**args*, ***kwargs*)
Return a new Signature bound to this app.

start(*argv=None*)

Run **celery** using *argv*.

Uses `sys.argv` if *argv* is not specified.

steps = None

Custom bootsteps to extend and modify the worker. See extending-bootsteps.

subclass_with_self(*Class, name=None, attribute='app', reverse=None, keep_reduce=False, **kw*)

Subclass an app-compatible class.

App-compatible means that the class has a class attribute that provides the default app it should use, for example: `class Foo: app = None`.

Parameters

- **Class** (*type*) – The app-compatible class to subclass.
- **name** (*str*) – Custom name for the target class.
- **attribute** (*str*) – Name of the attribute holding the app, Default is 'app'.
- **reverse** (*str*) – Reverse path to this object used for pickling purposes. For example, to get `app.AsyncResult`, use "AsyncResult".
- **keep_reduce** (*bool*) – If enabled a custom `__reduce__` implementation won't be provided.

task(**args, **opts*)

Decorator to create a task class out of any callable.

See Task options for a list of the arguments that can be passed to this decorator.

Examples

```
@app.task
def refresh_feed(url):
    store_feed(feedparser.parse(url))
```

with setting extra options:

```
@app.task(exchange='feeds')
def refresh_feed(url):
    return store_feed(feedparser.parse(url))
```

Note: App Binding: For custom apps the task decorator will return a proxy object, so that the act of creating the task is not performed until the task is used or the task registry is accessed.

If you're depending on binding to be deferred, then you must not access any attributes on the returned object until the application is fully set up (finalized).

task_cls = 'celery.app.task:Task'

tasks

Task registry.

Warning: Accessing this attribute will also auto-finalize the app.

property thread_oid

Per-thread unique identifier for this app.

timezone

Current timezone for this app.

This is a cached property taking the time zone from the **:setting:`timezone`** setting.

user_options = None

Custom options for command-line programs. See `extending-commandoptions`

uses_utc_timezone()

Check if the application uses the UTC timezone.

worker_main(argv=None)

Run **celery worker** using *argv*.

Uses `sys.argv` if *argv* is not specified.

`app.celery.make_celery(app: flask.app.Flask) → celery.app.base.Celery`

2.1.3 app.exceptions

Description

Module for managing exceptions.

References

flask-restx: <https://flask-restx.readthedocs.io/en/latest/errors.html>

Functions

init_app(app)

app.exceptions.init_app

`app.exceptions.init_app(app: flask.app.Flask) → None`

Exceptions

FileEmptyError

app.exceptions.FileEmptyError

exception app.exceptions.FileEmptyError

exception app.exceptions.FileEmptyError

args

characters_written

errno

POSIX exception code

filename

exception filename

filename2

second exception filename

strerror

exception strerror

with_traceback()

Exception.with_traceback(tb) – set self.__traceback__ to tb and return self.

app.exceptions._handle_validation_error_exception(*ex: marshmallow.exceptions.ValidationError*) →
tuple

app.exceptions.init_app(*app: flask.app.Flask*) → None

2.1.4 app.extensions

Description

Registers third party extensions.

Functions

init_app(app)

app.extensions.init_app

`app.extensions.init_app(app: flask.app.Flask) → None`

`app.extensions.init_app(app: flask.app.Flask) → None`

2.1.5 app.managers

Description

Registers database managers.

The managers are classes for managing database queries through database models.

Modules

`app.managers.base`

`app.managers.document`

`app.managers.role`

`app.managers.user`

app.managers.base

Description

Classes

`BaseManager(*args, **kwargs)`

app.managers.base.BaseManager

```
class app.managers.base.BaseManager(*args, **kwargs)
    Bases: object
```

Methods

BaseManager.__init__(*args, **kwargs)

BaseManager.create(**kwargs)

BaseManager.delete(record_id)

BaseManager.find(record_id, *args)

BaseManager.get(**kwargs)

BaseManager.raw(query)

BaseManager.save(record_id, **kwargs)

app.managers.base.BaseManager.__init__

`BaseManager.__init__(*args, **kwargs)`

app.managers.base.BaseManager.create

`BaseManager.create(**kwargs)`

app.managers.base.BaseManager.delete

`BaseManager.delete(record_id: int)`

app.managers.base.BaseManager.find

`BaseManager.find(record_id: int, *args)`

app.managers.base.BaseManager.get

`BaseManager.get(**kwargs)`

app.managers.base.BaseManager.raw

`BaseManager.raw(query: str)`

app.managers.base.BaseManager.save

`BaseManager.save(record_id: int, **kwargs)`

class `app.managers.base.BaseManager(*args, **kwargs)`

`create(**kwargs)`

`delete(record_id: int)`

`find(record_id: int, *args)`

`get(**kwargs)`

`raw(query: str)`

`save(record_id: int, **kwargs)`

app.managers.document**Description****Classes**

DocumentManager()

app.managers.document.DocumentManager

class `app.managers.document.DocumentManager`

Bases: `app.managers.base.BaseManager`

Methods

DocumentManager.__init__()

*DocumentManager.create(**kwargs)*

DocumentManager.delete(record_id)

*DocumentManager.find(record_id, *args)*

*DocumentManager.get(**kwargs)*

DocumentManager.raw(query)

*DocumentManager.save(record_id,
**kwargs)*

app.managers.document.DocumentManager.__init__

DocumentManager.**__init__**()

app.managers.document.DocumentManager.create

DocumentManager.**create**(**kwargs)

app.managers.document.DocumentManager.delete

DocumentManager.**delete**(record_id: int)

app.managers.document.DocumentManager.find

DocumentManager.**find**(record_id: int, *args)

app.managers.document.DocumentManager.get

DocumentManager.**get**(**kwargs)

app.managers.document.DocumentManager.raw

DocumentManager.**raw**(query: str)

app.managers.document.DocumentManager.save

DocumentManager.**save**(record_id: int, **kwargs)

class app.managers.document.DocumentManager

create(**kwargs)

delete(record_id: int)

find(record_id: int, *args)

get(**kwargs)

raw(query: str)

save(record_id: int, **kwargs)

app.managers.role**Description****Classes**

RoleManager()

app.managers.role.RoleManager**class** `app.managers.role.RoleManager`Bases: `app.managers.base.BaseManager`**Methods**

RoleManager.__init__()

*RoleManager.create(**kwargs)*

RoleManager.delete(record_id)

*RoleManager.find(record_id, *args)*

*RoleManager.get(**kwargs)*

RoleManager.raw(query)

*RoleManager.save(record_id, **kwargs)*

app.managers.role.RoleManager.__init__`RoleManager.__init__()`

app.managers.role.RoleManager.create`RoleManager.create(**kwargs)`**app.managers.role.RoleManager.delete**`RoleManager.delete(record_id: int)`**app.managers.role.RoleManager.find**`RoleManager.find(record_id: int, *args)`**app.managers.role.RoleManager.get**`RoleManager.get(**kwargs)`**app.managers.role.RoleManager.raw**`RoleManager.raw(query: str)`**app.managers.role.RoleManager.save**`RoleManager.save(record_id: int, **kwargs)`**class app.managers.role.RoleManager**`create(**kwargs)``delete(record_id: int)``find(record_id: int, *args)``get(**kwargs)``raw(query: str)``save(record_id: int, **kwargs)`**app.managers.user****Description****Classes**

`UserManager()`

app.managers.user.UserManager

class app.managers.user.**UserManager**
Bases: *app.managers.base.BaseManager*

Methods

userManager.__init__()

*userManager.create(**kwargs)*

userManager.delete(record_id)

*userManager.find(record_id, *args)*

*userManager.find_by_email(email, *args)*

*userManager.get(**kwargs)*

userManager.get_last_record()

userManager.raw(query)

*userManager.save(record_id, **kwargs)*

app.managers.user.UserManager.__init__

UserManager.**__init__**()

app.managers.user.UserManager.create

UserManager.**create**(**kwargs)

app.managers.user.UserManager.delete

`userManager.delete(record_id: int)`

app.managers.user.UserManager.find

`userManager.find(record_id: int, *args)`

app.managers.user.UserManager.find_by_email

`userManager.find_by_email(email: str, *args)`

app.managers.user.UserManager.get

`userManager.get(**kwargs)`

app.managers.user.UserManager.get_last_record

`userManager.get_last_record()`

app.managers.user.UserManager.raw

`userManager.raw(query: str)`

app.managers.user.UserManager.save

`userManager.save(record_id: int, **kwargs)`

class app.managers.user.UserManager

create(**kwargs)

delete(record_id: int)

find(record_id: int, *args)

find_by_email(email: str, *args)

get(**kwargs)

get_last_record()

raw(query: str)

save(record_id: int, **kwargs)

2.1.6 app.middleware

Description

WSGI middleware for validating requests content type.

Classes

<code>Middleware(app)</code>	WSGI middleware for checking if the request has a valid content type.
------------------------------	---

app.middleware.Middleware

class app.middleware.Middleware(*app: flask.app.Flask*)

Bases: object

WSGI middleware for checking if the request has a valid content type.

Methods

<code>Middleware.__init__(app)</code>

<code>Middleware.parse_content_type(content_type)</code>	Parser a request Content-Type.
--	--------------------------------

app.middleware.Middleware.__init__

Middleware.__init__(*app: flask.app.Flask*)

app.middleware.Middleware.parse_content_type

static Middleware.parse_content_type(*content_type: str*) → str

Parser a request Content-Type.

Parameters *content_type* (*str*) – Request Content Type.

Returns Parsed request Content Type.

Return type str

References

RFC 1341 - MIME (Multipurpose Internet Mail Extensions): <https://tools.ietf.org/html/rfc1341>

Examples

```
>>> from app.middleware import Middleware as m
>>> m.parse_content_type('multipart/form-data; boundary=something')
multipart/form-data
>>> m.parse_content_type('text/html; charset=utf-8')
text/html
```

class `app.middleware.Middleware`(*app*: `flask.app.Flask`)
WSGI middleware for checking if the request has a valid content type.

static `parse_content_type`(*content_type*: `str`) → `str`
Parser a request Content-Type.

Parameters `content_type` (`str`) – Request Content Type.

Returns Parsed request Content Type.

Return type `str`

References

RFC 1341 - MIME (Multipurpose Internet Mail Extensions): <https://tools.ietf.org/html/rfc1341>

Examples

```
>>> from app.middleware import Middleware as m
>>> m.parse_content_type('multipart/form-data; boundary=something')
multipart/form-data
>>> m.parse_content_type('text/html; charset=utf-8')
text/html
```

2.1.7 app.models

Description

Registers database models.

TODO: pending to define models with suffix “Model”.

There is not possible to rename the models with suffix “Model” because Flask-Security-Too doesn’t allow it. Maybe in the next major version could be available. <https://github.com/Flask-Middleware/flask-security/issues/395>

Modules

`app.models.base`

`app.models.document`

`app.models.role`

continues on next page

Table 83 – continued from previous page

<i>app.models.user</i>
<i>app.models.user_roles</i>

app.models.base

Description

Classes

<i>Base(*args, **kwargs)</i>

app.models.base.Base

class app.models.base.**Base**(*args, **kwargs)
Bases: playhouse.flask_utils.FlaskDB.get_model_class.<locals>.BaseModel

Attributes

<i>Base.dirty_fields</i>
<i>Base.id</i>

app.models.base.Base.dirty_fields

property Base.dirty_fields

app.models.base.Base.id

Base.id = <AutoField: Base.id>

Methods

<i>Base.__init__(*args, **kwargs)</i>
<i>Base.add_index(*fields, **kwargs)</i>
<i>Base.alias([alias])</i>
<i>Base.bind(database[, bind_refs, ...])</i>

continues on next page

Table 86 – continued from previous page

<i>Base.bind_ctx</i> (database[, bind_refs, ...])
<i>Base.bulk_create</i> (model_list[, batch_size])
<i>Base.bulk_update</i> (model_list, fields[, ...])
<i>Base.clone</i> ()
<i>Base.coerce</i> ([_coerce])
<i>Base.copy</i> (method)
<i>Base.create</i> (**query)
<i>Base.create_table</i> ([safe])
<i>Base.delete</i> ()
<i>Base.delete_by_id</i> (pk)
<i>Base.delete_instance</i> ([recursive, ...])
<i>Base.dependencies</i> ([search_nullable])
<i>Base.drop_table</i> ([safe, drop_sequences])
<i>Base.filter</i> (*dq_nodes, **filters)
<i>Base.get</i> (*query, **filters)
<i>Base.get_by_id</i> (pk)
<i>Base.get_fields</i> ([exclude, include, sort_order])
<i>Base.get_id</i> ()
<i>Base.get_or_create</i> (**kwargs)
<i>Base.get_or_none</i> (*query, **filters)
<i>Base.index</i> (*fields, **kwargs)
<i>Base.insert</i> ([_Model__data])
<i>Base.insert_from</i> (query, fields)
<i>Base.insert_many</i> (rows[, fields])
<i>Base.is_alias</i> ()

continues on next page

Table 86 – continued from previous page

<i>Base.is_dirty()</i>
<i>Base.noop()</i>
<i>Base.raw(query)</i>
<i>Base.reload()</i>
<i>Base.replace([_Model__data])</i>
<i>Base.replace_many(rows[, fields])</i>
<i>Base.save(*args, **kwargs)</i>
<i>Base.select(*fields)</i>
<i>Base.set_by_id(key, value)</i>
<i>Base.table_exists()</i>
<i>Base.truncate_table(**options)</i>
<i>Base.unwrap()</i>
<i>Base.update([_Model__data])</i>
<i>Base.validate_model()</i>

app.models.base.Base.__init__**Base.__init__**(*args, **kwargs)**app.models.base.Base.add_index****classmethod** **Base.add_index**(*fields, **kwargs)

app.models.base.Base.alias**classmethod** Base.**alias**(*alias=None*)**app.models.base.Base.bind****classmethod** Base.**bind**(*database, bind_refs=True, bind_backrefs=True, _exclude=None*)**app.models.base.Base.bind_ctx****classmethod** Base.**bind_ctx**(*database, bind_refs=True, bind_backrefs=True*)**app.models.base.Base.bulk_create****classmethod** Base.**bulk_create**(*model_list, batch_size=None*)**app.models.base.Base.bulk_update****classmethod** Base.**bulk_update**(*model_list, fields, batch_size=None*)**app.models.base.Base.clone**Base.**clone**()**app.models.base.Base.coerce**Base.**coerce**(*_coerce=True*)**app.models.base.Base.copy****static** Base.**copy**(*method*)**app.models.base.Base.create****classmethod** Base.**create**(***query*)

app.models.base.Base.create_table

```
classmethod Base.create_table(safe=True, **options)
```

app.models.base.Base.delete

```
classmethod Base.delete()
```

app.models.base.Base.delete_by_id

```
classmethod Base.delete_by_id(pk)
```

app.models.base.Base.delete_instance

```
Base.delete_instance(recursive=False, delete_nullable=False)
```

app.models.base.Base.dependencies

```
Base.dependencies(search_nullable=False)
```

app.models.base.Base.drop_table

```
classmethod Base.drop_table(safe=True, drop_sequences=True, **options)
```

app.models.base.Base.filter

```
classmethod Base.filter(*dq_nodes, **filters)
```

app.models.base.Base.get

```
classmethod Base.get(*query, **filters)
```

app.models.base.Base.get_by_id

```
classmethod Base.get_by_id(pk)
```

app.models.base.Base.get_fields

classmethod Base.get_fields(*exclude: Optional[list] = None, include: Optional[list] = None, sort_order: Optional[list] = None*) → set

app.models.base.Base.get_id

Base.get_id()

app.models.base.Base.get_or_create

classmethod Base.get_or_create(**kwargs)

app.models.base.Base.get_or_none

classmethod Base.get_or_none(*query, **filters)

app.models.base.Base.index

classmethod Base.index(*fields, **kwargs)

app.models.base.Base.insert

classmethod Base.insert(_Model__data=None, **insert)

app.models.base.Base.insert_from

classmethod Base.insert_from(query, fields)

app.models.base.Base.insert_many

classmethod Base.insert_many(rows, fields=None)

app.models.base.Base.is_alias

Base.is_alias()

app.models.base.Base.is_dirty`Base.is_dirty()`**app.models.base.Base.noop**`classmethod Base.noop()`**app.models.base.Base.raw**`static Base.raw(query: str)`**app.models.base.Base.reload**`Base.reload()`**app.models.base.Base.replace**`classmethod Base.replace(_Model__data=None, **insert)`**app.models.base.Base.replace_many**`classmethod Base.replace_many(rows, fields=None)`**app.models.base.Base.save**`abstract Base.save(*args: list, **kwargs: dict) → int`**app.models.base.Base.select**`classmethod Base.select(*fields)`**app.models.base.Base.set_by_id**`classmethod Base.set_by_id(key, value)`

app.models.base.Base.table_exists

classmethod Base.table_exists()

app.models.base.Base.truncate_table

classmethod Base.truncate_table(**options)

app.models.base.Base.unwrap

Base.unwrap()

app.models.base.Base.update

classmethod Base.update(_Model__data=None, **update)

app.models.base.Base.validate_model

classmethod Base.validate_model()

class app.models.base.Base(*args, **kwargs)

DoesNotExist

alias of app.models.base.BaseDoesNotExist

_coerce = True

_meta = <peewee.Metadata object>

classmethod _normalize_data(data, kwargs)

property _pk

_pk_expr()

_populate_unsaved_relations(field_dict)

_prune_fields(field_dict, only)

_schema = <peewee.SchemaManager object>

classmethod add_index(*fields, **kwargs)

classmethod alias(alias=None)

classmethod bind(database, bind_refs=True, bind_backrefs=True, _exclude=None)

classmethod bind_ctx(database, bind_refs=True, bind_backrefs=True)

classmethod bulk_create(model_list, batch_size=None)

classmethod bulk_update(model_list, fields, batch_size=None)

clone()

coerce(_coerce=True)

static copy(method)

```

classmethod create(**query)
classmethod create_table(safe=True, **options)
classmethod delete()
classmethod delete_by_id(pk)
delete_instance(recursive=False, delete_nullable=False)
dependencies(search_nullable=False)
property dirty_fields
classmethod drop_table(safe=True, drop_sequences=True, **options)
classmethod filter(*dq_nodes, **filters)
classmethod get(*query, **filters)
classmethod get_by_id(pk)
classmethod get_fields(exclude: Optional[list] = None, include: Optional[list] = None, sort_order:
                        Optional[list] = None) → set
get_id()
classmethod get_or_create(**kwargs)
classmethod get_or_none(*query, **filters)
id = <AutoField: Base.id>
classmethod index(*fields, **kwargs)
classmethod insert(_Model__data=None, **insert)
classmethod insert_from(query, fields)
classmethod insert_many(rows, fields=None)
is_alias()
is_dirty()
classmethod noop()
static raw(query: str)
reload()
classmethod replace(_Model__data=None, **insert)
classmethod replace_many(rows, fields=None)
abstract save(*args: list, **kwargs: dict) → int
classmethod select(*fields)
classmethod set_by_id(key, value)
classmethod table_exists()
classmethod truncate_table(**options)
unwrap()
classmethod update(_Model__data=None, **update)
classmethod validate_model()

```

app.models.document

Description

Classes

*Document(*args, **kwargs)*

app.models.document.Document

class app.models.document.**Document**(*args, **kwargs)
Bases: *app.models.base.Base*

Attributes

Document.created_at

Document.created_by

Document.created_by_id

Document.deleted_at

Document.directory_path

Document.dirty_fields

Document.id

Document.internal_filename

Document.mime_type

Document.name

Document.size

Document.updated_at

Document.url

app.models.document.Document.created_at

Document.created_at = <TimestampField: Document.created_at>

app.models.document.Document.created_by

Document.created_by = <ForeignKeyField: Document.created_by>

app.models.document.Document.created_by_id

Document.created_by_id = <ForeignKeyField: Document.created_by>

app.models.document.Document.deleted_at

Document.deleted_at = <TimestampField: Document.deleted_at>

app.models.document.Document.directory_path

Document.directory_path = <CharField: Document.directory_path>

app.models.document.Document.dirty_fields

property Document.dirty_fields

app.models.document.Document.id

Document.id = <AutoField: Document.id>

app.models.document.Document.internal_filename

Document.internal_filename = <CharField: Document.internal_filename>

app.models.document.Document.mime_type

Document.mime_type = <CharField: Document.mime_type>

app.models.document.Document.name

```
Document.name = <CharField: Document.name>
```

app.models.document.Document.size

```
Document.size = <IntegerField: Document.size>
```

app.models.document.Document.updated_at

```
Document.updated_at = <TimestampField: Document.updated_at>
```

app.models.document.Document.url

```
property Document.url
```

Methods

```
Document.__init__(*args, **kwargs)
```

```
Document.add_index(*fields, **kwargs)
```

```
Document.alias([alias])
```

```
Document.bind(database[, bind_refs, ...])
```

```
Document.bind_ctx(database[, bind_refs,  
...])
```

```
Document.bulk_create(model_list[,  
batch_size])
```

```
Document.bulk_update(model_list, fields[,  
...])
```

```
Document.clone()
```

```
Document.coerce([_coerce])
```

```
Document.copy(method)
```

```
Document.create(**query)
```

```
Document.create_table([safe])
```

```
Document.delete()
```

```
Document.delete_by_id(pk)
```

```
Document.delete_instance([recursive, ...])
```

continues on next page

Table 89 – continued from previous page

<code>Document.dependencies([search_nullable])</code>
<code>Document.drop_table([safe, drop_sequences])</code>
<code>Document.filter(*dq_nodes, **filters)</code>
<code>Document.get(*query, **filters)</code>
<code>Document.get_by_id(pk)</code>
<code>Document.get_fields([exclude, include, ...])</code>
<code>Document.get_filepath()</code>
<code>Document.get_id()</code>
<code>Document.get_or_create(**kwargs)</code>
<code>Document.get_or_none(*query, **filters)</code>
<code>Document.index(*fields, **kwargs)</code>
<code>Document.insert([_Model__data])</code>
<code>Document.insert_from(query, fields)</code>
<code>Document.insert_many(rows[, fields])</code>
<code>Document.is_alias()</code>
<code>Document.is_dirty()</code>
<code>Document.noop()</code>
<code>Document.raw(query)</code>
<code>Document.reload()</code>
<code>Document.replace([_Model__data])</code>
<code>Document.replace_many(rows[, fields])</code>
<code>Document.save(*args, **kwargs)</code>
<code>Document.select(*fields)</code>
<code>Document.set_by_id(key, value)</code>
<code>Document.table_exists()</code>

continues on next page

Table 89 – continued from previous page

*Document.truncate_table(**options)*

Document.unwrap()

Document.update([_Model__data])

Document.validate_model()

app.models.document.Document.__init__*Document.__init__(*args, **kwargs)***app.models.document.Document.add_index****classmethod** *Document.add_index(*fields, **kwargs)***app.models.document.Document.alias****classmethod** *Document.alias(alias=None)***app.models.document.Document.bind****classmethod** *Document.bind(database, bind_refs=True, bind_backrefs=True, _exclude=None)***app.models.document.Document.bind_ctx****classmethod** *Document.bind_ctx(database, bind_refs=True, bind_backrefs=True)***app.models.document.Document.bulk_create****classmethod** *Document.bulk_create(model_list, batch_size=None)***app.models.document.Document.bulk_update****classmethod** *Document.bulk_update(model_list, fields, batch_size=None)*

app.models.document.Document.clone

`Document.clone()`

app.models.document.Document.coerce

`Document.coerce(_coerce=True)`

app.models.document.Document.copy

`static Document.copy(method)`

app.models.document.Document.create

`classmethod Document.create(**query)`

app.models.document.Document.create_table

`classmethod Document.create_table(safe=True, **options)`

app.models.document.Document.delete

`classmethod Document.delete()`

app.models.document.Document.delete_by_id

`classmethod Document.delete_by_id(pk)`

app.models.document.Document.delete_instance

`Document.delete_instance(recursive=False, delete_nullable=False)`

app.models.document.Document.dependencies

`Document.dependencies(search_nullable=False)`

app.models.document.Document.drop_table

classmethod Document.**drop_table**(safe=True, drop_sequences=True, **options)

app.models.document.Document.filter

classmethod Document.**filter**(*dq_nodes, **filters)

app.models.document.Document.get

classmethod Document.**get**(*query, **filters)

app.models.document.Document.get_by_id

classmethod Document.**get_by_id**(pk)

app.models.document.Document.get_fields

classmethod Document.**get_fields**(exclude: Optional[list] = None, include: Optional[list] = None, sort_order: Optional[list] = None) → set

app.models.document.Document.get_filepath

Document.**get_filepath**()

app.models.document.Document.get_id

Document.**get_id**()

app.models.document.Document.get_or_create

classmethod Document.**get_or_create**(**kwargs)

app.models.document.Document.get_or_none

classmethod Document.**get_or_none**(*query, **filters)

app.models.document.Document.index

classmethod Document.**index**(*fields, **kwargs)

app.models.document.Document.insert

classmethod Document.**insert**(*_Model__data=None*, **insert)

app.models.document.Document.insert_from

classmethod Document.**insert_from**(query, fields)

app.models.document.Document.insert_many

classmethod Document.**insert_many**(rows, fields=None)

app.models.document.Document.is_alias

Document.**is_alias**()

app.models.document.Document.is_dirty

Document.**is_dirty**()

app.models.document.Document.noop

classmethod Document.**noop**()

app.models.document.Document.raw

static Document.**raw**(query: str)

app.models.document.Document.reload

Document.**reload**()

app.models.document.Document.replace**classmethod** Document.**replace**(*_Model__data=None, **insert*)**app.models.document.Document.replace_many****classmethod** Document.**replace_many**(*rows, fields=None*)**app.models.document.Document.save****abstract** Document.**save**(**args: list, **kwargs: dict*) → int**app.models.document.Document.select****classmethod** Document.**select**(**fields*)**app.models.document.Document.set_by_id****classmethod** Document.**set_by_id**(*key, value*)**app.models.document.Document.table_exists****classmethod** Document.**table_exists**()**app.models.document.Document.truncate_table****classmethod** Document.**truncate_table**(***options*)**app.models.document.Document.unwrap**Document.**unwrap**()**app.models.document.Document.update****classmethod** Document.**update**(*_Model__data=None, **update*)

app.models.document.Document.validate_model

```

    classmethod Document.validate_model()
class app.models.document.Document(*args, **kwargs)

    DoesNotExist
        alias of app.models.document.DocumentDoesNotExist
    _coerce = True
    _meta = <peewee.Metadata object>
    classmethod _normalize_data(data, kwargs)
    property _pk
    _pk_expr()
    _populate_unsaved_relations(field_dict)
    _prune_fields(field_dict, only)
    _schema = <peewee.SchemaManager object>
    classmethod add_index(*fields, **kwargs)
    classmethod alias(alias=None)
    classmethod bind(database, bind_refs=True, bind_backrefs=True, _exclude=None)
    classmethod bind_ctx(database, bind_refs=True, bind_backrefs=True)
    classmethod bulk_create(model_list, batch_size=None)
    classmethod bulk_update(model_list, fields, batch_size=None)
    clone()
    coerce(_coerce=True)
    static copy(method)
    classmethod create(**query)
    classmethod create_table(safe=True, **options)
    created_at = <TimestampField: Document.created_at>
    created_by = <ForeignKeyField: Document.created_by>
    created_by_id = <ForeignKeyField: Document.created_by>
    classmethod delete()
    classmethod delete_by_id(pk)
    delete_instance(recursive=False, delete_nullable=False)
    deleted_at = <TimestampField: Document.deleted_at>
    dependencies(search_nullable=False)
    directory_path = <CharField: Document.directory_path>
    property dirty_fields
    classmethod drop_table(safe=True, drop_sequences=True, **options)

```

```
classmethod filter(*dq_nodes, **filters)
classmethod get(*query, **filters)
classmethod get_by_id(pk)
classmethod get_fields(exclude: Optional[list] = None, include: Optional[list] = None, sort_order:
                        Optional[list] = None) → set

get_filepath()
get_id()
classmethod get_or_create(**kwargs)
classmethod get_or_none(*query, **filters)
id = <AutoField: Document.id>
classmethod index(*fields, **kwargs)
classmethod insert(_Model__data=None, **insert)
classmethod insert_from(query, fields)
classmethod insert_many(rows, fields=None)
internal_filename = <CharField: Document.internal_filename>
is_alias()
is_dirty()
mime_type = <CharField: Document.mime_type>
name = <CharField: Document.name>
classmethod noop()
static raw(query: str)
reload()
classmethod replace(_Model__data=None, **insert)
classmethod replace_many(rows, fields=None)
abstract save(*args: list, **kwargs: dict) → int
classmethod select(*fields)
classmethod set_by_id(key, value)
size = <IntegerField: Document.size>
classmethod table_exists()
classmethod truncate_table(**options)
unwrap()
classmethod update(_Model__data=None, **update)
updated_at = <TimestampField: Document.updated_at>
property url
classmethod validate_model()
```

app.models.role

Description

Classes

Role(*args, **kwargs)

app.models.role.Role

class app.models.role.**Role**(*args, **kwargs)

Bases: *app.models.base.Base*, flask_security.core.RoleMixin

Attributes

Role.created_at

Role.deleted_at

Role.description

Role.dirty_fields

Role.id

Role.label

Role.name

Role.roles

Role.updated_at

Role.userrolethrough_set

Role.users

app.models.role.Role.created_at

`Role.created_at = <TimestampField: Role.created_at>`

app.models.role.Role.deleted_at

`Role.deleted_at = <TimestampField: Role.deleted_at>`

app.models.role.Role.description

`Role.description = <TextField: Role.description>`

app.models.role.Role.dirty_fields

`property Role.dirty_fields`

app.models.role.Role.id

`Role.id = <AutoField: Role.id>`

app.models.role.Role.label

`Role.label = <CharField: Role.label>`

app.models.role.Role.name

`Role.name = <CharField: Role.name>`

app.models.role.Role.roles

`Role.roles`

app.models.role.Role.updated_at

`Role.updated_at = <TimestampField: Role.updated_at>`

app.models.role.Role.userrolethrough_set**Role.userrolethrough_set****app.models.role.Role.users****Role.users = <ManyToManyField: Role.users>****Methods***Role.__init__*(*args, **kwargs)*Role.add_index*(*fields, **kwargs)*Role.add_permissions*(permissions) Add one or more permissions to role.*Role.alias*([alias])*Role.bind*(database[, bind_refs, ...])*Role.bind_ctx*(database[, bind_refs, ...])*Role.bulk_create*(model_list[, batch_size])*Role.bulk_update*(model_list, fields[, ...])*Role.clone*()*Role.coerce*([_coerce])*Role.copy*(method)*Role.create*(**query)*Role.create_table*([safe])*Role.delete*()*Role.delete_by_id*(pk)*Role.delete_instance*([recursive, ...])*Role.dependencies*([search_nullable])*Role.drop_table*([safe, drop_sequences])*Role.filter*(*dq_nodes, **filters)*Role.get*(*query, **filters)

continues on next page

Table 92 – continued from previous page

<i>Role.get_by_id(pk)</i>	
<i>Role.get_fields</i> ([exclude, include, sort_order])	
<i>Role.get_id()</i>	
<i>Role.get_or_create</i> (**kwargs)	
<i>Role.get_or_none</i> (*query, **filters)	
<i>Role.get_permissions</i> ()	Return set of permissions associated with role.
<i>Role.index</i> (*fields, **kwargs)	
<i>Role.insert</i> ([_Model__data])	
<i>Role.insert_from</i> (query, fields)	
<i>Role.insert_many</i> (rows[, fields])	
<i>Role.is_alias</i> ()	
<i>Role.is_dirty</i> ()	
<i>Role.noop</i> ()	
<i>Role.raw</i> (query)	
<i>Role.reload</i> ()	
<i>Role.remove_permissions</i> (permissions)	Remove one or more permissions from role.
<i>Role.replace</i> ([_Model__data])	
<i>Role.replace_many</i> (rows[, fields])	
<i>Role.save</i> (*args, **kwargs)	
<i>Role.select</i> (*fields)	
<i>Role.set_by_id</i> (key, value)	
<i>Role.table_exists</i> ()	
<i>Role.truncate_table</i> (**options)	
<i>Role.unwrap</i> ()	
<i>Role.update</i> ([_Model__data])	
<i>Role.validate_model</i> ()	

app.models.role.Role.__init__

`Role.__init__(*args, **kwargs)`

app.models.role.Role.add_index

classmethod `Role.add_index(*fields, **kwargs)`

app.models.role.Role.add_permissions

`Role.add_permissions(permissions: Union[set, list, str]) → None`

Add one or more permissions to role.

Parameters **permissions** – a set, list, or single string.

New in version 3.3.0.

Deprecated since version 3.4.4: Use `UserDatastore.add_permissions_to_role()`

app.models.role.Role.alias

classmethod `Role.alias(alias=None)`

app.models.role.Role.bind

classmethod `Role.bind(database, bind_refs=True, bind_backrefs=True, _exclude=None)`

app.models.role.Role.bind_ctx

classmethod `Role.bind_ctx(database, bind_refs=True, bind_backrefs=True)`

app.models.role.Role.bulk_create

classmethod `Role.bulk_create(model_list, batch_size=None)`

app.models.role.Role.bulk_update

classmethod `Role.bulk_update(model_list, fields, batch_size=None)`

app.models.role.Role.clone

`Role.clone()`

app.models.role.Role.coerce

`Role.coerce(_coerce=True)`

app.models.role.Role.copy

`static Role.copy(method)`

app.models.role.Role.create

`classmethod Role.create(**query)`

app.models.role.Role.create_table

`classmethod Role.create_table(safe=True, **options)`

app.models.role.Role.delete

`classmethod Role.delete()`

app.models.role.Role.delete_by_id

`classmethod Role.delete_by_id(pk)`

app.models.role.Role.delete_instance

`Role.delete_instance(recursive=False, delete_nullable=False)`

app.models.role.Role.dependencies

`Role.dependencies(search_nullable=False)`

app.models.role.Role.drop_table

classmethod `Role.drop_table(safe=True, drop_sequences=True, **options)`

app.models.role.Role.filter

classmethod `Role.filter(*dq_nodes, **filters)`

app.models.role.Role.get

classmethod `Role.get(*query, **filters)`

app.models.role.Role.get_by_id

classmethod `Role.get_by_id(pk)`

app.models.role.Role.get_fields

classmethod `Role.get_fields(exclude: Optional[list] = None, include: Optional[list] = None, sort_order: Optional[list] = None) → set`

app.models.role.Role.get_id

`Role.get_id()`

app.models.role.Role.get_or_create

classmethod `Role.get_or_create(**kwargs)`

app.models.role.Role.get_or_none

classmethod `Role.get_or_none(*query, **filters)`

app.models.role.Role.get_permissions

`Role.get_permissions() → set`

Return set of permissions associated with role.

Supports permissions being a comma separated string, an iterable, or a set based on how the underlying DB model was built.

New in version 3.3.0.

app.models.role.Role.index

```
classmethod Role.index(*fields, **kwargs)
```

app.models.role.Role.insert

```
classmethod Role.insert(_Model__data=None, **insert)
```

app.models.role.Role.insert_from

```
classmethod Role.insert_from(query, fields)
```

app.models.role.Role.insert_many

```
classmethod Role.insert_many(rows, fields=None)
```

app.models.role.Role.is_alias

```
Role.is_alias()
```

app.models.role.Role.is_dirty

```
Role.is_dirty()
```

app.models.role.Role.noop

```
classmethod Role.noop()
```

app.models.role.Role.raw

```
static Role.raw(query: str)
```

app.models.role.Role.reload

```
Role.reload()
```

app.models.role.Role.remove_permissions

Role.remove_permissions(permissions: Union[set, list, str]) → None

Remove one or more permissions from role.

Parameters **permissions** – a set, list, or single string.

New in version 3.3.0.

Deprecated since version 3.4.4: Use `UserDatastore.remove_permissions_from_role()`

app.models.role.Role.replace

classmethod **Role.replace**(*_Model__data=None, **insert*)

app.models.role.Role.replace_many

classmethod **Role.replace_many**(rows, fields=None)

app.models.role.Role.save

abstract **Role.save**(*args: list, **kwargs: dict) → int

app.models.role.Role.select

classmethod **Role.select**(*fields)

app.models.role.Role.set_by_id

classmethod **Role.set_by_id**(key, value)

app.models.role.Role.table_exists

classmethod **Role.table_exists**()

app.models.role.Role.truncate_table

classmethod **Role.truncate_table**(**options)

app.models.role.Role.unwrap

`Role.unwrap()`

app.models.role.Role.update

`classmethod Role.update(_Model__data=None, **update)`

app.models.role.Role.validate_model

`classmethod Role.validate_model()`

`class app.models.role.Role(*args, **kwargs)`

DoesNotExist

alias of `app.models.role.RoleDoesNotExist`

`_coerce = True`

`_meta = <peewee.Metadata object>`

`classmethod _normalize_data(data, kwargs)`

`property _pk`

`_pk_expr()`

`_populate_unsaved_relations(field_dict)`

`_prune_fields(field_dict, only)`

`_schema = <peewee.SchemaManager object>`

`classmethod add_index(*fields, **kwargs)`

`add_permissions(permissions: Union[set, list, str]) → None`

Add one or more permissions to role.

Parameters `permissions` – a set, list, or single string.

New in version 3.3.0.

Deprecated since version 3.4.4: Use `UserDatastore.add_permissions_to_role()`

`classmethod alias(alias=None)`

`classmethod bind(database, bind_refs=True, bind_backrefs=True, _exclude=None)`

`classmethod bind_ctx(database, bind_refs=True, bind_backrefs=True)`

`classmethod bulk_create(model_list, batch_size=None)`

`classmethod bulk_update(model_list, fields, batch_size=None)`

`clone()`

`coerce(_coerce=True)`

`static copy(method)`

`classmethod create(**query)`

`classmethod create_table(safe=True, **options)`


```

created_at = <TimestampField: Role.created_at>
classmethod delete()
classmethod delete_by_id(pk)
delete_instance(recursive=False, delete_nullable=False)
deleted_at = <TimestampField: Role.deleted_at>
dependencies(search_nullable=False)
description = <TextField: Role.description>
property dirty_fields
classmethod drop_table(safe=True, drop_sequences=True, **options)
classmethod filter(*dq_nodes, **filters)
classmethod get(*query, **filters)
classmethod get_by_id(pk)
classmethod get_fields(exclude: Optional[list] = None, include: Optional[list] = None, sort_order:
                        Optional[list] = None) → set

get_id()
classmethod get_or_create(**kwargs)
classmethod get_or_none(*query, **filters)
get_permissions() → set
    Return set of permissions associated with role.

    Supports permissions being a comma separated string, an iterable, or a set based on how the underlying
    DB model was built.

    New in version 3.3.0.

id = <AutoField: Role.id>
classmethod index(*fields, **kwargs)
classmethod insert(_Model__data=None, **insert)
classmethod insert_from(query, fields)
classmethod insert_many(rows, fields=None)
is_alias()
is_dirty()
label = <CharField: Role.label>
name = <CharField: Role.name>
classmethod noop()
static raw(query: str)
reload()
remove_permissions(permissions: Union[set, list, str]) → None
    Remove one or more permissions from role.

    Parameters permissions – a set, list, or single string.

```

New in version 3.3.0.

Deprecated since version 3.4.4: Use `UserDatastore.remove_permissions_from_role()`

```
classmethod replace(_Model__data=None, **insert)
classmethod replace_many(rows, fields=None)
roles
abstract save(*args: list, **kwargs: dict) → int
classmethod select(*fields)
classmethod set_by_id(key, value)
classmethod table_exists()
classmethod truncate_table(**options)
unwrap()
classmethod update(_Model__data=None, **update)
updated_at = <TimestampField: Role.updated_at>
userrolethrough_set
users = <ManyToManyField: Role.users>
classmethod validate_model()
```

app.models.user

Description

Classes

<code>User(*args, **kwargs)</code>	User database model.
------------------------------------	----------------------

app.models.user.User

```
class app.models.user.User(*args, **kwargs)
    Bases: app.models.base.Base, flask_security.core.UserMixin
    User database model.
```

References

fs_uniquifier field is required by flask-security-too: <https://flask-security-too.readthedocs.io/en/stable/changelog.html#version-4-0-0>

Attributes

<i>User.active</i>	
<i>User.birth_date</i>	
<i>User.children</i>	
<i>User.created_at</i>	
<i>User.created_by</i>	
<i>User.created_by_id</i>	
<i>User.deleted_at</i>	
<i>User.dirty_fields</i>	
<i>User.document_set</i>	
<i>User.email</i>	
<i>User.fs_uniquifier</i>	
<i>User.genre</i>	
<i>User.id</i>	
<i>User.is_active</i>	Returns <i>True</i> if the user is active.
<i>User.is_anonymous</i>	
<i>User.is_authenticated</i>	
<i>User.last_name</i>	
<i>User.name</i>	
<i>User.password</i>	
<i>User.roles</i>	
<i>User.updated_at</i>	
<i>User.userrolethrough_set</i>	

app.models.user.User.active

User.active = <BooleanField: User.active>

app.models.user.User.birth_date

User.birth_date = <DateField: User.birth_date>

app.models.user.User.children

User.children

app.models.user.User.created_at

User.created_at = <TimestampField: User.created_at>

app.models.user.User.created_by

User.created_by = <ForeignKeyField: User.created_by>

app.models.user.User.created_by_id

User.created_by_id = <ForeignKeyField: User.created_by>

app.models.user.User.deleted_at

User.deleted_at = <TimestampField: User.deleted_at>

app.models.user.User.dirty_fields

property User.dirty_fields

app.models.user.User.document_set

User.document_set

app.models.user.User.email

```
User.email = <CharField: User.email>
```

app.models.user.User.fs_uniquifier

```
User.fs_uniquifier = <TextField: User.fs_uniquifier>
```

app.models.user.User.genre

```
User.genre = <FixedCharField: User.genre>
```

app.models.user.User.id

```
User.id = <AutoField: User.id>
```

app.models.user.User.is_active

```
property User.is_active: bool
    Returns True if the user is active.
```

app.models.user.User.is_anonymous

```
property User.is_anonymous
```

app.models.user.User.is_authenticated

```
property User.is_authenticated
```

app.models.user.User.last_name

```
User.last_name = <CharField: User.last_name>
```

app.models.user.User.name

```
User.name = <CharField: User.name>
```

app.models.user.User.password

User.password = <CharField: User.password>

app.models.user.User.roles

User.roles = <ManyToManyField: User.roles>

app.models.user.User.updated_at

User.updated_at = <TimestampField: User.updated_at>

app.models.user.User.userrolethrough_set

User.userrolethrough_set

Methods

User.__init__(*args, **kwargs)

User.add_index(*fields, **kwargs)

User.alias([alias])

User.bind(database[, bind_refs, ...])

User.bind_ctx(database[, bind_refs, ...])

User.bulk_create(model_list[, batch_size])

User.bulk_update(model_list, fields[, ...])

<i>User.calc_username</i> ()	Come up with the best 'username' based on how the app is configured (via SECURITY_USER_IDENTITY_ATTRIBUTES).
------------------------------	--

User.clone()

User.coerce([_coerce])

User.copy(method)

User.create(**query)

User.create_table([safe])

User.delete()

continues on next page

Table 95 – continued from previous page

<code>User.delete_by_id(pk)</code>	
<code>User.delete_instance([recursive, ...])</code>	
<code>User.dependencies([search_nullable])</code>	
<code>User.drop_table([safe, drop_sequences])</code>	
<code>User.ensure_password(plain_text)</code>	
<code>User.filter(*dq_nodes, **filters)</code>	
<code>User.get(*query, **filters)</code>	
<code>User.get_auth_token()</code>	Constructs the user's authentication token.
<code>User.get_by_id(pk)</code>	
<code>User.get_fields([exclude, include, sort_order])</code>	
<code>User.get_id()</code>	Returns the user identification attribute.
<code>User.get_or_create(**kwargs)</code>	
<code>User.get_or_none(*query, **filters)</code>	
<code>User.get_redirect_qparams([existing])</code>	Return user info that will be added to redirect query params.
<code>User.get_reset_token()</code>	
<code>User.get_security_payload()</code>	Serialize user object as response payload.
<code>User.has_permission(permission)</code>	Returns <i>True</i> if user has this permission (via a role it has).
<code>User.has_role(role)</code>	Returns <i>True</i> if the user identifies with the specified role.
<code>User.index(*fields, **kwargs)</code>	
<code>User.insert([_Model__data])</code>	
<code>User.insert_from(query, fields)</code>	
<code>User.insert_many(rows[, fields])</code>	
<code>User.is_alias()</code>	
<code>User.is_dirty()</code>	
<code>User.noop()</code>	
<code>User.raw(query)</code>	
<code>User.reload()</code>	

continues on next page

Table 95 – continued from previous page

<code>User.replace([_Model__data])</code>	
<code>User.replace_many(rows[, fields])</code>	
<code>User.save(*args, **kwargs)</code>	
<code>User.select(*fields)</code>	
<code>User.set_by_id(key, value)</code>	
<code>User.table_exists()</code>	
<code>User.tf_send_security_token(method, **kwargs)</code>	Generate and send the security code for two-factor.
<code>User.truncate_table(**options)</code>	
<code>User.unwrap()</code>	
<code>User.update([_Model__data])</code>	
<code>User.us_send_security_token(method, **kwargs)</code>	Generate and send the security code for unified sign in.
<code>User.validate_model()</code>	
<code>User.verify_and_update_password(password)</code>	Returns True if the password is valid for the specified user.
<code>User.verify_auth_token(data)</code>	Perform additional verification of contents of auth token.
<code>User.verify_reset_token(token)</code>	

app.models.user.User.__init__

`User.__init__(*args, **kwargs)`

app.models.user.User.add_index

`classmethod User.add_index(*fields, **kwargs)`

app.models.user.User.alias

`classmethod User.alias(alias=None)`

app.models.user.User.bind

classmethod User.**bind**(*database, bind_refs=True, bind_backrefs=True, _exclude=None*)

app.models.user.User.bind_ctx

classmethod User.**bind_ctx**(*database, bind_refs=True, bind_backrefs=True*)

app.models.user.User.bulk_create

classmethod User.**bulk_create**(*model_list, batch_size=None*)

app.models.user.User.bulk_update

classmethod User.**bulk_update**(*model_list, fields, batch_size=None*)

app.models.user.User.calc_username

User.**calc_username**() → str

Come up with the best ‘username’ based on how the app is configured (via SECURITY_USER_IDENTITY_ATTRIBUTES). Returns the first non-null match (and converts to string). In theory this should NEVER be the empty string unless the user record isn’t actually valid.

New in version 3.4.0.

app.models.user.User.clone

User.**clone**()

app.models.user.User.coerce

User.**coerce**(*_coerce=True*)

app.models.user.User.copy

static User.**copy**(*method*)

app.models.user.User.create

classmethod User.create(***query*)

app.models.user.User.create_table

classmethod User.create_table(*safe=True*, ***options*)

app.models.user.User.delete

classmethod User.delete()

app.models.user.User.delete_by_id

classmethod User.delete_by_id(*pk*)

app.models.user.User.delete_instance

User.delete_instance(*recursive=False*, *delete_nullable=False*)

app.models.user.User.dependencies

User.dependencies(*search_nullable=False*)

app.models.user.User.drop_table

classmethod User.drop_table(*safe=True*, *drop_sequences=True*, ***options*)

app.models.user.User.ensure_password

static User.ensure_password(*plain_text: str*) → str

app.models.user.User.filter

classmethod User.filter(**dq_nodes*, ***filters*)

app.models.user.User.get

classmethod `User.get(*query, **filters)`

app.models.user.User.get_auth_token

`User.get_auth_token()` → Union[str, bytes]

Constructs the user's authentication token.

Raises ValueError – If `fs_token_uniquifier` is part of model but not set.

Optionally use a separate uniquifier so that changing password doesn't invalidate auth tokens.

This data **MUST** be securely signed using the `remember_token_serializer`

Changed in version 4.0.0: If user model has `fs_token_uniquifier` - use that (raise `ValueError` if not set). Otherwise fallback to using `fs_uniquifier`.

app.models.user.User.get_by_id

classmethod `User.get_by_id(pk)`

app.models.user.User.get_fields

classmethod `User.get_fields(exclude: Optional[list] = None, include: Optional[list] = None, sort_order: Optional[list] = None) → set`

app.models.user.User.get_id

`User.get_id()`

Returns the user identification attribute. 'Alternative-token' for Flask-Login. This is always `fs_uniquifier`.

New in version 3.4.0.

app.models.user.User.get_or_create

classmethod `User.get_or_create(**kwargs)`

app.models.user.User.get_or_none

classmethod `User.get_or_none(*query, **filters)`

app.models.user.User.get_redirect_qparams

`User.get_redirect_qparams(existing: Optional[Dict[str, Any]] = None) → Dict[str, Any]`

Return user info that will be added to redirect query params.

Parameters **existing** – A dict that will be updated.

Returns A dict whose keys will be query params and values will be query values.

The returned dict will always have an ‘identity’ key/value. If the User Model contains ‘email’, an ‘email’ key/value will added. All keys provided in ‘existing’ will also be merged in.

New in version 3.2.0.

Changed in version 4.0.0: Add ‘identity’ using UserMixin.calc_username() - email is optional.

app.models.user.User.get_reset_token

`User.get_reset_token() → str`

app.models.user.User.get_security_payload

`User.get_security_payload() → Dict[str, Any]`

Serialize user object as response payload. Override this to return any/all of the user object in JSON responses. Return a dict.

app.models.user.User.has_permission

`User.has_permission(permission: str) → bool`

Returns *True* if user has this permission (via a role it has).

Parameters **permission** – permission string name

New in version 3.3.0.

app.models.user.User.has_role

`User.has_role(role: Union[str, Role]) → bool`

Returns *True* if the user identifies with the specified role.

Parameters **role** – A role name or *Role* instance

app.models.user.User.index

`classmethod User.index(*fields, **kwargs)`

app.models.user.User.insert

```
classmethod User.insert(_Model__data=None, **insert)
```

app.models.user.User.insert_from

```
classmethod User.insert_from(query, fields)
```

app.models.user.User.insert_many

```
classmethod User.insert_many(rows, fields=None)
```

app.models.user.User.is_alias

```
User.is_alias()
```

app.models.user.User.is_dirty

```
User.is_dirty()
```

app.models.user.User.noop

```
classmethod User.noop()
```

app.models.user.User.raw

```
static User.raw(query: str)
```

app.models.user.User.reload

```
User.reload()
```

app.models.user.User.replace

```
classmethod User.replace(_Model__data=None, **insert)
```

app.models.user.User.replace_many

classmethod `User.replace_many(rows, fields=None)`

app.models.user.User.save

`User.save(*args: list, **kwargs: dict) → int`

app.models.user.User.select

classmethod `User.select(*fields)`

app.models.user.User.set_by_id

classmethod `User.set_by_id(key, value)`

app.models.user.User.table_exists

classmethod `User.table_exists()`

app.models.user.User.tf_send_security_token

`User.tf_send_security_token(method: str, **kwargs: Any) → Optional[str]`

Generate and send the security code for two-factor.

Parameters

- **method** – The method in which the code will be sent
- **kwargs** – Opaque parameters that are subject to change at any time

Returns None if successful, error message if not.

This is a wrapper around `tf_send_security_token()` that can be overridden to manage any errors.

New in version 3.4.0.

app.models.user.User.truncate_table

classmethod `User.truncate_table(**options)`

app.models.user.User.unwrap

`User.unwrap()`

app.models.user.User.update

```
classmethod User.update(_Model__data=None, **update)
```

app.models.user.User.us_send_security_token

```
User.us_send_security_token(method: str, **kwargs: Any) → Optional[str]
```

Generate and send the security code for unified sign in.

Parameters

- **method** – The method in which the code will be sent
- **kwargs** – Opaque parameters that are subject to change at any time

Returns None if successful, error message if not.

This is a wrapper around `us_send_security_token()` that can be overridden to manage any errors.

New in version 3.4.0.

app.models.user.User.validate_model

```
classmethod User.validate_model()
```

app.models.user.User.verify_and_update_password

```
User.verify_and_update_password(password: str) → bool
```

Returns True if the password is valid for the specified user.

Additionally, the hashed password in the database is updated if the hashing algorithm happens to have changed.

N.B. you MUST call DB commit if you are using a session-based datastore (such as SQLAlchemy) since the user instance might have been altered (i.e. `app.security.datastore.commit()`). This is usually handled in the view.

Parameters **password** – A plaintext password to verify

New in version 3.2.0.

app.models.user.User.verify_auth_token

```
User.verify_auth_token(data: Union[str, bytes]) → bool
```

Perform additional verification of contents of auth token. Prior to this being called the token has been validated (via signing) and has not expired.

Parameters **data** – the data as formulated by `get_auth_token()`

New in version 3.3.0.

Changed in version 4.0.0: If user model has `fs_token_uniquifier` - use that otherwise use `fs_uniquifier`.

app.models.user.User.verify_reset_token**static** User.verify_reset_token(token: str) → any**class** app.models.user.User(*args, **kwargs)

User database model.

Referencesfs_uniquifier field is required by flask-security-too: <https://flask-security-too.readthedocs.io/en/stable/changelog.html#version-4-0-0>**DoesNotExist**

alias of app.models.user.UserDoesNotExist

_coerce = True**_meta** = <peewee.Metadata object>**classmethod** _normalize_data(data, kwargs)**property** _pk**_pk_expr**()**_populate_unsaved_relations**(field_dict)**_prune_fields**(field_dict, only)**_schema** = <peewee.SchemaManager object>**active** = <BooleanField: User.active>**classmethod** add_index(*fields, **kwargs)**classmethod** alias(alias=None)**classmethod** bind(database, bind_refs=True, bind_backrefs=True, _exclude=None)**classmethod** bind_ctx(database, bind_refs=True, bind_backrefs=True)**birth_date** = <DateField: User.birth_date>**classmethod** bulk_create(model_list, batch_size=None)**classmethod** bulk_update(model_list, fields, batch_size=None)**calc_username**() → str

Come up with the best 'username' based on how the app is configured (via SECURITY_USER_IDENTITY_ATTRIBUTES). Returns the first non-null match (and converts to string). In theory this should NEVER be the empty string unless the user record isn't actually valid.

New in version 3.4.0.

children**clone**()**coerce**(_coerce=True)**static** copy(method)**classmethod** create(**query)**classmethod** create_table(safe=True, **options)


```

created_at = <TimestampField: User.created_at>
created_by = <ForeignKeyField: User.created_by>
created_by_id = <ForeignKeyField: User.created_by>
classmethod delete()
classmethod delete_by_id(pk)
delete_instance(recursive=False, delete_nullable=False)
deleted_at = <TimestampField: User.deleted_at>
dependencies(search_nullable=False)
property dirty_fields
document_set
classmethod drop_table(safe=True, drop_sequences=True, **options)
email = <CharField: User.email>
static ensure_password(plain_text: str) → str
classmethod filter(*dq_nodes, **filters)
fs_uniquifier = <TextField: User.fs_uniquifier>
genre = <FixedCharField: User.genre>
classmethod get(*query, **filters)
get_auth_token() → Union[str, bytes]
    Constructs the user's authentication token.

    Raises ValueError – If fs_token_uniquifier is part of model but not set.

    Optionally use a separate uniquifier so that changing password doesn't invalidate auth tokens.

    This data MUST be securely signed using the remember_token_serializer

    Changed in version 4.0.0: If user model has fs_token_uniquifier - use that (raise ValueError if not
    set). Otherwise fallback to using fs_uniquifier.

classmethod get_by_id(pk)
classmethod get_fields(exclude: Optional[list] = None, include: Optional[list] = None, sort_order:
    Optional[list] = None) → set

get_id()
    Returns the user identification attribute. 'Alternative-token' for Flask-Login. This is always
    fs_uniquifier.

    New in version 3.4.0.

classmethod get_or_create(**kwargs)
classmethod get_or_none(*query, **filters)
get_redirect_qparams(existing: Optional[Dict[str, Any]] = None) → Dict[str, Any]
    Return user info that will be added to redirect query params.

    Parameters existing – A dict that will be updated.

    Returns A dict whose keys will be query params and values will be query values.

```

The returned dict will always have an 'identity' key/value. If the User Model contains 'email', an 'email' key/value will added. All keys provided in 'existing' will also be merged in.

New in version 3.2.0.

Changed in version 4.0.0: Add 'identity' using UserMixin.calc_username() - email is optional.

get_reset_token() → str

get_security_payload() → Dict[str, Any]

Serialize user object as response payload. Override this to return any/all of the user object in JSON responses. Return a dict.

has_permission(permission: str) → bool

Returns *True* if user has this permission (via a role it has).

Parameters permission – permission string name

New in version 3.3.0.

has_role(role: Union[str, Role]) → bool

Returns *True* if the user identifies with the specified role.

Parameters role – A role name or *Role* instance

id = <AutoField: User.id>

classmethod index(*fields, **kwargs)

classmethod insert(_Model__data=None, **insert)

classmethod insert_from(query, fields)

classmethod insert_many(rows, fields=None)

property is_active: bool

Returns *True* if the user is active.

is_alias()

property is_anonymous

property is_authenticated

is_dirty()

last_name = <CharField: User.last_name>

name = <CharField: User.name>

classmethod noop()

password = <CharField: User.password>

static raw(query: str)

reload()

classmethod replace(_Model__data=None, **insert)

classmethod replace_many(rows, fields=None)

roles = <ManyToManyField: User.roles>

save(*args: list, **kwargs: dict) → int

classmethod select(*fields)

classmethod set_by_id(key, value)

classmethod `table_exists()`

tf_send_security_token(*method: str, **kwargs: Any*) → Optional[str]
Generate and send the security code for two-factor.

Parameters

- **method** – The method in which the code will be sent
- **kwargs** – Opaque parameters that are subject to change at any time

Returns None if successful, error message if not.

This is a wrapper around `tf_send_security_token()` that can be overridden to manage any errors.

New in version 3.4.0.

classmethod `truncate_table(**options)`

unwrap()

classmethod `update(_Model__data=None, **update)`

updated_at = <TimestampField: User.updated_at>

us_send_security_token(*method: str, **kwargs: Any*) → Optional[str]
Generate and send the security code for unified sign in.

Parameters

- **method** – The method in which the code will be sent
- **kwargs** – Opaque parameters that are subject to change at any time

Returns None if successful, error message if not.

This is a wrapper around `us_send_security_token()` that can be overridden to manage any errors.

New in version 3.4.0.

userrolethrough_set

classmethod `validate_model()`

verify_and_update_password(*password: str*) → bool
Returns True if the password is valid for the specified user.

Additionally, the hashed password in the database is updated if the hashing algorithm happens to have changed.

N.B. you MUST call DB commit if you are using a session-based datastore (such as SQLAlchemy) since the user instance might have been altered (i.e. `app.security.datastore.commit()`). This is usually handled in the view.

Parameters **password** – A plaintext password to verify

New in version 3.2.0.

verify_auth_token(*data: Union[str, bytes]*) → bool

Perform additional verification of contents of auth token. Prior to this being called the token has been validated (via signing) and has not expired.

Parameters **data** – the data as formulated by `get_auth_token()`

New in version 3.3.0.

Changed in version 4.0.0: If user model has `fs_token_uniquifier` - use that otherwise use `fs_uniquifier`.

```
static verify_reset_token(token: str) → any
```

app.models.user_roles

Description

Functions

```
get_db_models()
```

app.models.get_db_models

```
app.models.get_db_models() → list
```

```
app.models.get_db_models() → list
```

2.1.8 app.serializers

Description

Modules for managing data from requests and responses.

Serializers are modules based on Marshmallow.

Marshmallow is an ORM/ODM/framework-agnostic library for converting complex datatypes, such as objects, to and from native Python datatypes.

In short, marshmallow schemas can be used to:

- Validate input data.
- Deserialize input data to app-level objects.
- Serialize app-level objects to primitive Python types. The serialized objects can then be rendered to standard formats such as JSON for use in an HTTP API.

References

Pre-/Post-processor Invocation Order: <https://marshmallow.readthedocs.io/en/stable/extending.html?highlight=step1#pre-post-processor-invocation-order>

Modules

```
app.serializers.auth
```

```
app.serializers.core
```

```
app.serializers.document
```

continues on next page

Table 97 – continued from previous page

`app.serializers.role`

`app.serializers.user`

app.serializers.auth**Description****Classes**

`AuthUserConfirmResetPasswordSerializer(*[,
...])`

`AuthUserLoginSerializer(*[, only, exclude, ...])`

app.serializers.auth.AuthUserConfirmResetPasswordSerializer

```
class app.serializers.auth.AuthUserConfirmResetPasswordSerializer(*, only: Op-  
tional[Union[Sequence[str],  
Set[str]]] = None, exclude:  
Union[Sequence[str],  
Set[str]] = (), many: bool =  
False, context: Optional[Dict]  
= None, load_only:  
Union[Sequence[str],  
Set[str]] = (), dump_only:  
Union[Sequence[str],  
Set[str]] = (), partial:  
Union[bool, Sequence[str],  
Set[str]] = False, unknown:  
Optional[str] = None)
```

Bases: flask_marshmallow.schema.Schema

Attributes

`AuthUserConfirmResetPasswordSerializer.
TYPE_MAPPING`

`AuthUserConfirmResetPasswordSerializer.
dict_class`

`AuthUserConfirmResetPasswordSerializer`Overrides for default schema-level error mes-
`error_messages`sages

`AuthUserConfirmResetPasswordSerializer.
opts`

`AuthUserConfirmResetPasswordSerializer.
set_class`

app.serializers.auth.AuthUserConfirmResetPasswordSerializer.TYPE_MAPPING

```
AuthUserConfirmResetPasswordSerializer.TYPE_MAPPING = {<class 'str':>:
<class 'marshmallow.fields.String'>, <class 'bytes':>: <class
'marshmallow.fields.String'>, <class 'datetime.datetime':>: <class
'marshmallow.fields.DateTime'>, <class 'float':>: <class
'marshmallow.fields.Float'>, <class 'bool':>: <class
'marshmallow.fields.Boolean'>, <class 'tuple':>: <class
'marshmallow.fields.Raw'>, <class 'list':>: <class
'marshmallow.fields.Raw'>, <class 'set':>: <class 'marshmallow.fields.Raw'>,
<class 'int':>: <class 'marshmallow.fields.Integer'>, <class 'uuid.UUID':>:
<class 'marshmallow.fields.UUID'>, <class 'datetime.time':>: <class
'marshmallow.fields.Time'>, <class 'datetime.date':>: <class
'marshmallow.fields.Date'>, <class 'datetime.timedelta':>: <class
'marshmallow.fields.TimeDelta'>, <class 'decimal.Decimal':>: <class
'marshmallow.fields.Decimal'>}
```

app.serializers.auth.AuthUserConfirmResetPasswordSerializer.dict_class

property AuthUserConfirmResetPasswordSerializer.dict_class: type

app.serializers.auth.AuthUserConfirmResetPasswordSerializer.error_messages

```
AuthUserConfirmResetPasswordSerializer.error_messages = {}
    Overrides for default schema-level error messages
```

app.serializers.auth.AuthUserConfirmResetPasswordSerializer.opts

AuthUserConfirmResetPasswordSerializer.opts = <marshmallow.schema.SchemaOpts object>

app.serializers.auth.AuthUserConfirmResetPasswordSerializer.set_class

property AuthUserConfirmResetPasswordSerializer.set_class: type

Methods

<i>AuthUserConfirmResetPasswordSerializer.</i>	
<i>__init__</i> (*)	
<i>AuthUserConfirmResetPasswordSerializer</i>	Serialize an object to native Python data types
<i>dump</i> (obj, *)	according to this Schema's fields.
<i>AuthUserConfirmResetPasswordSerializer</i>	Same as <i>dump()</i> , except return a JSON-
<i>dumps</i> (...)	encoded string.
<i>AuthUserConfirmResetPasswordSerializer</i>	Generate a <i>Schema</i> class given a dictionary of
<i>from_dict</i> (...)	fields.
<i>AuthUserConfirmResetPasswordSerializer</i>	Defines how to pull values from an object to se-
<i>get_attribute</i> (...)	rialize.

continues on next page

Table 100 – continued from previous page

<code>AuthUserConfirmResetPasswordSerializer.handle_error(...)</code>	Custom error handler function for the schema.
<code>AuthUserConfirmResetPasswordSerializer.jsonify(obj)</code>	Return a JSON response containing the serialized data.
<code>AuthUserConfirmResetPasswordSerializer.load(data, *)</code>	Deserialize a data structure to an object defined by this Schema's fields.
<code>AuthUserConfirmResetPasswordSerializer.loads(...)</code>	Same as <code>load()</code> , except it takes a JSON string as input.
<code>AuthUserConfirmResetPasswordSerializer.make_object(...)</code>	
<code>AuthUserConfirmResetPasswordSerializer.on_bind_field(...)</code>	Hook to modify a field when it is bound to the <i>Schema</i> .
<code>AuthUserConfirmResetPasswordSerializer.validate(data, *)</code>	Validate <i>data</i> against the schema, returning a dictionary of validation errors.
<code>AuthUserConfirmResetPasswordSerializer.validate_token(token)</code>	

`app.serializers.auth.AuthUserConfirmResetPasswordSerializer.__init__`

`AuthUserConfirmResetPasswordSerializer.__init__`(* , only: *Optional[Union[Sequence[str], Set[str]] = None, exclude: Union[Sequence[str], Set[str]] = (), many: bool = False, context: Optional[Dict] = None, load_only: Union[Sequence[str], Set[str]] = (), dump_only: Union[Sequence[str], Set[str]] = (), partial: Union[bool, Sequence[str], Set[str]] = False, unknown: Optional[str] = None)*

`app.serializers.auth.AuthUserConfirmResetPasswordSerializer.dump`

`AuthUserConfirmResetPasswordSerializer.dump`(*obj: Any, *, many: Optional[bool] = None*)

Serialize an object to native Python data types according to this Schema's fields.

Parameters

- **obj** – The object to serialize.
- **many** – Whether to serialize *obj* as a collection. If *None*, the value for *self.many* is used.

Returns

Serialized data

Changed in version 3.0.0b7: This method returns the serialized data rather than a (data, errors) tuple. A `ValidationError` is raised if *obj* is invalid.

Changed in version 3.0.0rc9: Validation no longer occurs upon serialization.

app.serializers.auth.AuthUserConfirmResetPasswordSerializer.dumps

`AuthUserConfirmResetPasswordSerializer.dumps(obj: Any, *args, many: Optional[bool] = None, **kwargs)`

Same as `dump()`, except return a JSON-encoded string.

Parameters

- **obj** – The object to serialize.
- **many** – Whether to serialize *obj* as a collection. If *None*, the value for *self.many* is used.

Returns A json string

New in version 1.0.0.

Changed in version 3.0.0b7: This method returns the serialized data rather than a (data, errors) tuple. A `ValidationError` is raised if *obj* is invalid.

app.serializers.auth.AuthUserConfirmResetPasswordSerializer.from_dict

classmethod `AuthUserConfirmResetPasswordSerializer.from_dict(fields: Dict[str, Union[marshmallow.fields.Field, type]], *, name: str = 'Generated-Schema') → type`

Generate a *Schema* class given a dictionary of fields.

```
from marshmallow import Schema, fields

PersonSchema = Schema.from_dict({"name": fields.Str()})
print(PersonSchema().load({"name": "David"})) # => {'name': 'David'}
```

Generated schemas are not added to the class registry and therefore cannot be referred to by name in *Nested* fields.

Parameters

- **fields** (*dict*) – Dictionary mapping field names to field instances.
- **name** (*str*) – Optional name for the class, which will appear in the `repr` for the class.

New in version 3.0.0.

app.serializers.auth.AuthUserConfirmResetPasswordSerializer.get_attribute

`AuthUserConfirmResetPasswordSerializer.get_attribute(obj: Any, attr: str, default: Any)`

Defines how to pull values from an object to serialize.

New in version 2.0.0.

Changed in version 3.0.0a1: Changed position of *obj* and *attr*.

app.serializers.auth.AuthUserConfirmResetPasswordSerializer.handle_error

`AuthUserConfirmResetPasswordSerializer.handle_error`(*error*: *marshmallow.exceptions.ValidationError*,
data: *Any*, *, *many*: *bool*,
***kwargs*)

Custom error handler function for the schema.

Parameters

- **error** – The *ValidationError* raised during (de)serialization.
- **data** – The original input data.
- **many** – Value of *many* on dump or load.
- **partial** – Value of *partial* on load.

New in version 2.0.0.

Changed in version 3.0.0rc9: Receives *many* and *partial* (on deserialization) as keyword arguments.

app.serializers.auth.AuthUserConfirmResetPasswordSerializer.jsonify

`AuthUserConfirmResetPasswordSerializer.jsonify`(*obj*, *many*=<*object object*>, **args*,
***kwargs*)

Return a JSON response containing the serialized data.

Parameters

- **obj** – Object to serialize.
- **many** (*bool*) – Whether *obj* should be serialized as an instance or as a collection. If unset, defaults to the value of the *many* attribute on this Schema.
- **kwargs** – Additional keyword arguments passed to *flask.jsonify*.

Changed in version 0.6.0: Takes the same arguments as *marshmallow.Schema.dump*. Additional keyword arguments are passed to *flask.jsonify*.

Changed in version 0.6.3: The *many* argument for this method defaults to the value of the *many* attribute on the Schema. Previously, the *many* argument of this method defaulted to *False*, regardless of the value of *Schema.many*.

app.serializers.auth.AuthUserConfirmResetPasswordSerializer.load

`AuthUserConfirmResetPasswordSerializer.load`(*data*: *Union[Mapping[str, Any], Iterable[Mapping[str, Any]]]*, *, *many*:
Optional[bool] = *None*, *partial*:
Optional[Union[bool, Sequence[str], Set[str]]] = *None*, *unknown*:
Optional[str] = *None*)

Deserialize a data structure to an object defined by this Schema's fields.

Parameters

- **data** – The data to deserialize.
- **many** – Whether to deserialize *data* as a collection. If *None*, the value for *self.many* is used.
- **partial** – Whether to ignore missing fields and not require any fields declared. Propagates down to *Nested* fields as well. If its value is an iterable, only missing fields listed in that iterable will be ignored. Use dot delimiters to specify nested fields.

- **unknown** – Whether to exclude, include, or raise an error for unknown fields in the data. Use *EXCLUDE*, *INCLUDE* or *RAISE*. If *None*, the value for *self.unknown* is used.

Returns Deserialized data

New in version 1.0.0.

Changed in version 3.0.0b7: This method returns the deserialized data rather than a (data, errors) tuple. A `ValidationError` is raised if invalid data are passed.

app.serializers.auth.AuthUserConfirmResetPasswordSerializer.loads

```
AuthUserConfirmResetPasswordSerializer.loads(json_data: str, *, many: Optional[bool]
                                              = None, partial: Optional[Union[bool,
                                              Sequence[str], Set[str]]] = None,
                                              unknown: Optional[str] = None,
                                              **kwargs)
```

Same as `load()`, except it takes a JSON string as input.

Parameters

- **json_data** – A JSON string of the data to deserialize.
- **many** – Whether to deserialize *obj* as a collection. If *None*, the value for *self.many* is used.
- **partial** – Whether to ignore missing fields and not require any fields declared. Propagates down to *Nested* fields as well. If its value is an iterable, only missing fields listed in that iterable will be ignored. Use dot delimiters to specify nested fields.
- **unknown** – Whether to exclude, include, or raise an error for unknown fields in the data. Use *EXCLUDE*, *INCLUDE* or *RAISE*. If *None*, the value for *self.unknown* is used.

Returns Deserialized data

New in version 1.0.0.

Changed in version 3.0.0b7: This method returns the deserialized data rather than a (data, errors) tuple. A `ValidationError` is raised if invalid data are passed.

app.serializers.auth.AuthUserConfirmResetPasswordSerializer.make_object

```
AuthUserConfirmResetPasswordSerializer.make_object(data, **kwargs)
```

app.serializers.auth.AuthUserConfirmResetPasswordSerializer.on_bind_field

```
AuthUserConfirmResetPasswordSerializer.on_bind_field(field_name: str, field_obj:
                                                    marshmallow.fields.Field) →
                                                    None
```

Hook to modify a field when it is bound to the *Schema*.

No-op by default.

app.serializers.auth.AuthUserConfirmResetPasswordSerializer.validate

```
AuthUserConfirmResetPasswordSerializer.validate(data: Union[Mapping[str, Any],
                                                         Iterable[Mapping[str, Any]]], *,
                                                         many: Optional[bool] = None,
                                                         partial: Optional[Union[bool,
                                                         Sequence[str], Set[str]]] = None)
                                                         → Dict[str, List[str]]
```

Validate *data* against the schema, returning a dictionary of validation errors.

Parameters

- **data** – The data to validate.
- **many** – Whether to validate *data* as a collection. If *None*, the value for *self.many* is used.
- **partial** – Whether to ignore missing fields and not require any fields declared. Propagates down to *Nested* fields as well. If its value is an iterable, only missing fields listed in that iterable will be ignored. Use dot delimiters to specify nested fields.

Returns A dictionary of validation errors.

New in version 1.1.0.

app.serializers.auth.AuthUserConfirmResetPasswordSerializer.validate_token

```
AuthUserConfirmResetPasswordSerializer.validate_token(token)
```

app.serializers.auth.AuthUserLoginSerializer

```
class app.serializers.auth.AuthUserLoginSerializer(*, only: Optional[Union[Sequence[str], Set[str]]]
                                                         = None, exclude: Union[Sequence[str], Set[str]]
                                                         = (), many: bool = False, context: Optional[Dict]
                                                         = None, load_only: Union[Sequence[str],
                                                         Set[str]] = (), dump_only: Union[Sequence[str],
                                                         Set[str]] = (), partial: Union[bool, Sequence[str],
                                                         Set[str]] = False, unknown: Optional[str] =
                                                         None)
```

Bases: flask_marshmallow.schema.Schema

Attributes

```
AuthUserLoginSerializer.
```

```
TYPE_MAPPING
```

```
AuthUserLoginSerializer.dict_class
```

```
AuthUserLoginSerializer.
```

```
error_messages
```

Overrides for default schema-level error messages

```
AuthUserLoginSerializer.opts
```

```
AuthUserLoginSerializer.set_class
```

app.serializers.auth.AuthUserLoginSerializer.TYPE_MAPPING

```
AuthUserLoginSerializer.TYPE_MAPPING = {<class 'str'>: <class
'marshmallow.fields.String'>, <class 'bytes'>: <class
'marshmallow.fields.String'>, <class 'datetime.datetime'>: <class
'marshmallow.fields.DateTime'>, <class 'float'>: <class
'marshmallow.fields.Float'>, <class 'bool'>: <class
'marshmallow.fields.Boolean'>, <class 'tuple'>: <class
'marshmallow.fields.Raw'>, <class 'list'>: <class
'marshmallow.fields.Raw'>, <class 'set'>: <class 'marshmallow.fields.Raw'>,
<class 'int'>: <class 'marshmallow.fields.Integer'>, <class 'uuid.UUID'>:
<class 'marshmallow.fields.UUID'>, <class 'datetime.time'>: <class
'marshmallow.fields.Time'>, <class 'datetime.date'>: <class
'marshmallow.fields.Date'>, <class 'datetime.timedelta'>: <class
'marshmallow.fields.TimeDelta'>, <class 'decimal.Decimal'>: <class
'marshmallow.fields.Decimal'>}
```

app.serializers.auth.AuthUserLoginSerializer.dict_class

property AuthUserLoginSerializer.dict_class: type

app.serializers.auth.AuthUserLoginSerializer.error_messages

AuthUserLoginSerializer.error_messages = {}
 Overrides for default schema-level error messages

app.serializers.auth.AuthUserLoginSerializer.opts

AuthUserLoginSerializer.opts = <marshmallow.schema.SchemaOpts object>

app.serializers.auth.AuthUserLoginSerializer.set_class

property AuthUserLoginSerializer.set_class: type

Methods

<code>AuthUserLoginSerializer.__init__(*[, only, ...])</code>	
<code>AuthUserLoginSerializer.dump(obj, *[, many])</code>	Serialize an object to native Python data types according to this Schema's fields.
<code>AuthUserLoginSerializer.dumps(obj, *args[, many])</code>	Same as <code>dump()</code> , except return a JSON-encoded string.
<code>AuthUserLoginSerializer.from_dict(fields, *)</code>	Generate a <i>Schema</i> class given a dictionary of fields.
<code>AuthUserLoginSerializer.get_attribute(obj, ...)</code>	Defines how to pull values from an object to serialize.

continues on next page

Table 102 – continued from previous page

<code>AuthUserLoginSerializer.handle_error(error, ...)</code>	Custom error handler function for the schema.
<code>AuthUserLoginSerializer.jsonify(obj[, many])</code>	Return a JSON response containing the serialized data.
<code>AuthUserLoginSerializer.load(data, *[, ...])</code>	Deserialize a data structure to an object defined by this Schema's fields.
<code>AuthUserLoginSerializer.loads(json_data, *)</code>	Same as <code>load()</code> , except it takes a JSON string as input.
<code>AuthUserLoginSerializer.make_object(data, ...)</code>	
<code>AuthUserLoginSerializer.on_bind_field(...)</code>	Hook to modify a field when it is bound to the <i>Schema</i> .
<code>AuthUserLoginSerializer.validate(data, *[, ...])</code>	Validate <i>data</i> against the schema, returning a dictionary of validation errors.
<code>AuthUserLoginSerializer.validate_email(email)</code>	
<code>AuthUserLoginSerializer.validate_password(...)</code>	

app.serializers.auth.AuthUserLoginSerializer.__init__

```
AuthUserLoginSerializer.__init__(*, only: Optional[Union[Sequence[str], Set[str]]] =
    None, exclude: Union[Sequence[str], Set[str]] = (),
    many: bool = False, context: Optional[Dict] = None,
    load_only: Union[Sequence[str], Set[str]] = (),
    dump_only: Union[Sequence[str], Set[str]] = (),
    partial: Union[bool, Sequence[str], Set[str]] = False,
    unknown: Optional[str] = None)
```

app.serializers.auth.AuthUserLoginSerializer.dump

`AuthUserLoginSerializer.dump(obj: Any, *, many: Optional[bool] = None)`
 Serialize an object to native Python data types according to this Schema's fields.

Parameters

- **obj** – The object to serialize.
- **many** – Whether to serialize *obj* as a collection. If *None*, the value for *self.many* is used.

Returns Serialized data

New in version 1.0.0.

Changed in version 3.0.0b7: This method returns the serialized data rather than a (data, errors) tuple. A `ValidationError` is raised if *obj* is invalid.

Changed in version 3.0.0rc9: Validation no longer occurs upon serialization.

app.serializers.auth.AuthUserLoginSerializer.dumps

`AuthUserLoginSerializer.dumps(obj: Any, *args, many: Optional[bool] = None, **kwargs)`
Same as `dump()`, except return a JSON-encoded string.

Parameters

- **obj** – The object to serialize.
- **many** – Whether to serialize *obj* as a collection. If *None*, the value for *self.many* is used.

Returns A json string

New in version 1.0.0.

Changed in version 3.0.0b7: This method returns the serialized data rather than a (data, errors) tuple. A `ValidationError` is raised if *obj* is invalid.

app.serializers.auth.AuthUserLoginSerializer.from_dict

classmethod `AuthUserLoginSerializer.from_dict(fields: Dict[str, Union[marshmallow.fields.Field, type]], *, name: str = 'GeneratedSchema') → type`

Generate a *Schema* class given a dictionary of fields.

```
from marshmallow import Schema, fields

PersonSchema = Schema.from_dict({"name": fields.Str()})
print(PersonSchema().load({"name": "David"})) # => {'name': 'David'}
```

Generated schemas are not added to the class registry and therefore cannot be referred to by name in *Nested* fields.

Parameters

- **fields** (*dict*) – Dictionary mapping field names to field instances.
- **name** (*str*) – Optional name for the class, which will appear in the `repr` for the class.

New in version 3.0.0.

app.serializers.auth.AuthUserLoginSerializer.get_attribute

`AuthUserLoginSerializer.get_attribute(obj: Any, attr: str, default: Any)`

Defines how to pull values from an object to serialize.

New in version 2.0.0.

Changed in version 3.0.0a1: Changed position of *obj* and *attr*.

app.serializers.auth.AuthUserLoginSerializer.handle_error

`AuthUserLoginSerializer.handle_error`(*error: marshmallow.exceptions.ValidationError*,
data: Any, *, *many: bool*, ***kwargs*)

Custom error handler function for the schema.

Parameters

- **error** – The *ValidationError* raised during (de)serialization.
- **data** – The original input data.
- **many** – Value of *many* on dump or load.
- **partial** – Value of *partial* on load.

New in version 2.0.0.

Changed in version 3.0.0rc9: Receives *many* and *partial* (on deserialization) as keyword arguments.

app.serializers.auth.AuthUserLoginSerializer.jsonify

`AuthUserLoginSerializer.jsonify`(*obj*, *many=<object object>*, **args*, ***kwargs*)

Return a JSON response containing the serialized data.

Parameters

- **obj** – Object to serialize.
- **many** (*bool*) – Whether *obj* should be serialized as an instance or as a collection. If unset, defaults to the value of the *many* attribute on this Schema.
- **kwargs** – Additional keyword arguments passed to *flask jsonify*.

Changed in version 0.6.0: Takes the same arguments as *marshmallow.Schema.dump*. Additional keyword arguments are passed to *flask jsonify*.

Changed in version 0.6.3: The *many* argument for this method defaults to the value of the *many* attribute on the Schema. Previously, the *many* argument of this method defaulted to *False*, regardless of the value of *Schema.many*.

app.serializers.auth.AuthUserLoginSerializer.load

`AuthUserLoginSerializer.load`(*data: Union[Mapping[str, Any], Iterable[Mapping[str, Any]]]*, *, *many: Optional[bool] = None*, *partial: Optional[Union[bool, Sequence[str], Set[str]]] = None*, *unknown: Optional[str] = None*)

Deserialize a data structure to an object defined by this Schema's fields.

Parameters

- **data** – The data to deserialize.
- **many** – Whether to deserialize *data* as a collection. If *None*, the value for *self.many* is used.
- **partial** – Whether to ignore missing fields and not require any fields declared. Propagates down to *Nested* fields as well. If its value is an iterable, only missing fields listed in that iterable will be ignored. Use dot delimiters to specify nested fields.
- **unknown** – Whether to exclude, include, or raise an error for unknown fields in the data. Use *EXCLUDE*, *INCLUDE* or *RAISE*. If *None*, the value for *self.unknown* is used.

Returns Deserialized data

New in version 1.0.0.

Changed in version 3.0.0b7: This method returns the deserialized data rather than a (data, errors) tuple. A `ValidationError` is raised if invalid data are passed.

`app.serializers.auth.AuthUserLoginSerializer.loads`

`AuthUserLoginSerializer.loads`(*json_data*: str, *, *many*: Optional[bool] = None, *partial*: Optional[Union[bool, Sequence[str], Set[str]]] = None, *unknown*: Optional[str] = None, **kwargs)

Same as `load()`, except it takes a JSON string as input.

Parameters

- **json_data** – A JSON string of the data to deserialize.
- **many** – Whether to deserialize *obj* as a collection. If *None*, the value for *self.many* is used.
- **partial** – Whether to ignore missing fields and not require any fields declared. Propagates down to *Nested* fields as well. If its value is an iterable, only missing fields listed in that iterable will be ignored. Use dot delimiters to specify nested fields.
- **unknown** – Whether to exclude, include, or raise an error for unknown fields in the data. Use *EXCLUDE*, *INCLUDE* or *RAISE*. If *None*, the value for *self.unknown* is used.

Returns Deserialized data

New in version 1.0.0.

Changed in version 3.0.0b7: This method returns the deserialized data rather than a (data, errors) tuple. A `ValidationError` is raised if invalid data are passed.

`app.serializers.auth.AuthUserLoginSerializer.make_object`

`AuthUserLoginSerializer.make_object`(*data*, **kwargs)

`app.serializers.auth.AuthUserLoginSerializer.on_bind_field`

`AuthUserLoginSerializer.on_bind_field`(*field_name*: str, *field_obj*: *marshmallow.fields.Field*) → None

Hook to modify a field when it is bound to the *Schema*.

No-op by default.

`app.serializers.auth.AuthUserLoginSerializer.validate`

`AuthUserLoginSerializer.validate`(*data*: Union[Mapping[str, Any], Iterable[Mapping[str, Any]]], *, *many*: Optional[bool] = None, *partial*: Optional[Union[bool, Sequence[str], Set[str]]] = None) → Dict[str, List[str]]

Validate *data* against the schema, returning a dictionary of validation errors.

Parameters

- **data** – The data to validate.
- **many** – Whether to validate *data* as a collection. If *None*, the value for *self.many* is used.

- **partial** – Whether to ignore missing fields and not require any fields declared. Propagates down to **Nested** fields as well. If its value is an iterable, only missing fields listed in that iterable will be ignored. Use dot delimiters to specify nested fields.

Returns A dictionary of validation errors.

New in version 1.1.0.

`app.serializers.auth.AuthUserLoginSerializer.validate_email`

`AuthUserLoginSerializer.validate_email(email)`

`app.serializers.auth.AuthUserLoginSerializer.validate_password`

`AuthUserLoginSerializer.validate_password(password)`

```
class app.serializers.auth.AuthUserConfirmResetPasswordSerializer(*, only: Optional[Union[Sequence[str], Set[str]]] = None, exclude: Union[Sequence[str], Set[str]] = (), many: bool = False, context: Optional[Dict] = None, load_only: Union[Sequence[str], Set[str]] = (), dump_only: Union[Sequence[str], Set[str]] = (), partial: Union[bool, Sequence[str], Set[str]] = False, unknown: Optional[str] = None)
```

`class Meta`

Options object for a Schema.

Example usage:

```
class Meta:
    fields = ("id", "email", "date_created")
    exclude = ("password", "secret_attribute")
```

Available options:

- **fields:** Tuple or list of fields to include in the serialized result.
- **additional:** Tuple or list of fields to include *in addition to the* explicitly declared fields. **additional** and **fields** are mutually-exclusive options.
- **include:** Dictionary of additional fields to include in the schema. It is usually better to define fields as class variables, but you may need to use this option, e.g., if your fields are Python keywords. May be an *OrderedDict*.
- **exclude:** Tuple or list of fields to exclude in the serialized result. Nested fields can be represented with dot delimiters.
- **dateformat:** Default format for *Date* <fields.Date> fields.
- **datetimeformat:** Default format for *DateTime* <fields.DateTime> fields.

- **timeformat:** Default format for *Time* <fields.Time> fields.
- **render_module:** Module to use for *loads* <Schema.loads> and *dumps* <Schema.dumps>. Defaults to *json* from the standard library.
- **ordered:** If *True*, order serialization output according to the order in which fields were declared. Output of *Schema.dump* will be a *collections.OrderedDict*.
- **index_errors:** If *True*, errors dictionaries will include the **index** of invalid items in a collection.
- **load_only:** Tuple or list of fields to exclude from serialized results.
- **dump_only:** Tuple or list of fields to exclude from deserialization
- **unknown:** Whether to exclude, include, or raise an error for unknown fields in the data. Use *EXCLUDE*, *INCLUDE* or *RAISE*.
- **register:** Whether to register the *Schema* with marshmallow's internal class registry. Must be *True* if you intend to refer to this *Schema* by class name in *Nested* fields. Only set this to *False* when memory usage is critical. Defaults to *True*.

OPTIONS_CLASS

alias of `marshmallow.schema.SchemaOpts`

```
TYPE_MAPPING = {<class 'str':>: <class 'marshmallow.fields.String'>, <class  
'bytes':>: <class 'marshmallow.fields.String'>, <class 'datetime.datetime':>: <class  
'marshmallow.fields.DateTime'>, <class 'float':>: <class  
'marshmallow.fields.Float'>, <class 'bool':>: <class 'marshmallow.fields.Boolean'>,  
<class 'tuple':>: <class 'marshmallow.fields.Raw'>, <class 'list':>: <class  
'marshmallow.fields.Raw'>, <class 'set':>: <class 'marshmallow.fields.Raw'>, <class  
'int':>: <class 'marshmallow.fields.Integer'>, <class 'uuid.UUID':>: <class  
'marshmallow.fields.UUID'>, <class 'datetime.time':>: <class  
'marshmallow.fields.Time'>, <class 'datetime.date':>: <class  
'marshmallow.fields.Date'>, <class 'datetime.timedelta':>: <class  
'marshmallow.fields.TimeDelta'>, <class 'decimal.Decimal':>: <class  
'marshmallow.fields.Decimal'>}
```

_bind_field(*field_name*: str, *field_obj*: marshmallow.fields.Field) → None

Bind field to the schema, setting any necessary attributes on the field (e.g. parent and name).

Also set field *load_only* and *dump_only* values if *field_name* was specified in class *Meta*.

static _call_and_store(*getter_func*, *data*, *, *field_name*, *error_store*, *index=None*)

Call *getter_func* with *data* as its argument, and store any *ValidationErrors*.

Parameters

- **getter_func** (*callable*) – Function for getting the serialized/deserialized value from *data*.
- **data** – The data passed to *getter_func*.
- **field_name** (*str*) – Field name.
- **index** (*int*) – Index of the item being validated, if validating a collection, otherwise *None*.

```

_declared_fields = {'password': <fields.String(dump_default=<marshmallow.missing>,
attribute=None, validate=<Length(min=8, max=50, equal=None, error=None)>,
required=True, load_only=True, dump_only=False, load_default=<marshmallow.missing>,
allow_none=False, error_messages={'required': 'Missing data for required field.',
'null': 'Field may not be null.', 'validator_failed': 'Invalid value.', 'invalid':
'Not a valid string.', 'invalid_utf8': 'Not a valid utf-8 string.'})>, 'token':
<fields.String(dump_default=<marshmallow.missing>, attribute=None, validate=None,
required=True, load_only=False, dump_only=False, load_default=<marshmallow.missing>,
allow_none=False, error_messages={'required': 'Missing data for required field.',
'null': 'Field may not be null.', 'validator_failed': 'Invalid value.', 'invalid':
'Not a valid string.', 'invalid_utf8': 'Not a valid utf-8 string.'})>>}

```

```

_default_error_messages = {'type': 'Invalid input type.', 'unknown': 'Unknown
field.'}

```

```

_deserialize(data: Union[Mapping[str, Any], Iterable[Mapping[str, Any]]], *, error_store:
    marshmallow.error_store.ErrorStore, many: bool = False, partial=False, unknown='raise',
    index=None) → Union[marshmallow.schema._T, List[marshmallow.schema._T]]

```

Deserialize data.

Parameters

- **data** (*dict*) – The data to deserialize.
- **error_store** (*ErrorStore*) – Structure to store errors.
- **many** (*bool*) – *True* if data should be deserialized as a collection.
- **partial** (*bool | tuple*) – Whether to ignore missing fields and not require any fields declared. Propagates down to Nested fields as well. If its value is an iterable, only missing fields listed in that iterable will be ignored. Use dot delimiters to specify nested fields.
- **unknown** – Whether to exclude, include, or raise an error for unknown fields in the data. Use *EXCLUDE*, *INCLUDE* or *RAISE*.
- **index** (*int*) – Index of the item being serialized (for storing errors) if serializing a collection, otherwise *None*.

Returns A dictionary of the deserialized data.

```

_do_load(data: Union[Mapping[str, Any], Iterable[Mapping[str, Any]]], *, many: Optional[bool] = None,
    partial: Optional[Union[bool, Sequence[str], Set[str]]] = None, unknown: Optional[str] = None,
    postprocess: bool = True)

```

Deserialize *data*, returning the deserialized result. This method is private API.

Parameters

- **data** – The data to deserialize.
- **many** – Whether to deserialize *data* as a collection. If *None*, the value for *self.many* is used.
- **partial** – Whether to validate required fields. If its value is an iterable, only fields listed in that iterable will be ignored will be allowed missing. If *True*, all fields will be allowed missing. If *None*, the value for *self.partial* is used.
- **unknown** – Whether to exclude, include, or raise an error for unknown fields in the data. Use *EXCLUDE*, *INCLUDE* or *RAISE*. If *None*, the value for *self.unknown* is used.
- **postprocess** – Whether to run *post_load* methods..

Returns Deserialized data

```

_has_processors(tag) → bool

```

```
_hooks = defaultdict(<class 'list'>, {('post_load', False): ['make_object'],
'validates': ['validate_token']})

_init_fields() → None
    Update self.fields, self.load_fields, and self.dump_fields based on schema options. This method is private
    API.

_invoke_dump_processors(tag: str, data, *, many: bool, original_data=None)

_invoke_field_validators(*, error_store: marshmallow.error_store.ErrorStore, data, many: bool)

_invoke_load_processors(tag: str, data, *, many: bool, original_data, partial: Union[bool,
    Sequence[str], Set[str]])

_invoke_processors(tag: str, *, pass_many: bool, data, many: bool, original_data=None, **kwargs)

_invoke_schema_validators(*, error_store: marshmallow.error_store.ErrorStore, pass_many: bool, data,
    original_data, many: bool, partial: Union[bool, Sequence[str], Set[str]],
    field_errors: bool = False)

_normalize_nested_options() → None
    Apply then flatten nested schema options. This method is private API.

_run_validator(validator_func, output, *, original_data, error_store, many, partial, pass_original,
    index=None)

_serialize(obj: Union[marshmallow.schema._T, Iterable[marshmallow.schema._T]], *, many: bool =
    False)
    Serialize obj.
```

Parameters

- **obj** – The object(s) to serialize.
- **many** (*bool*) – *True* if data should be serialized as a collection.

Returns A dictionary of the serialized data

Changed in version 1.0.0: Renamed from `marshal`.

property dict_class: *type*

dump(obj: Any, *, many: Optional[bool] = None)

Serialize an object to native Python data types according to this Schema's fields.

Parameters

- **obj** – The object to serialize.
- **many** – Whether to serialize *obj* as a collection. If *None*, the value for *self.many* is used.

Returns Serialized data

New in version 1.0.0.

Changed in version 3.0.0b7: This method returns the serialized data rather than a (*data*, *errors*) tuple. A `ValidationError` is raised if *obj* is invalid.

Changed in version 3.0.0rc9: Validation no longer occurs upon serialization.

dumps(obj: Any, *args, many: Optional[bool] = None, **kwargs)

Same as `dump()`, except return a JSON-encoded string.

Parameters

- **obj** – The object to serialize.

- **many** – Whether to serialize *obj* as a collection. If *None*, the value for *self.many* is used.

Returns A json string

New in version 1.0.0.

Changed in version 3.0.0b7: This method returns the serialized data rather than a (data, errors) tuple. A *ValidationError* is raised if *obj* is invalid.

error_messages = {}

Overrides for default schema-level error messages

fields

Dictionary mapping field_names -> Field objects

classmethod from_dict(fields: Dict[str, Union[mashmallow.fields.Field, type]], *, name: str = 'GeneratedSchema') → type

Generate a *Schema* class given a dictionary of fields.

```
from marshmallow import Schema, fields

PersonSchema = Schema.from_dict({"name": fields.Str()})
print(PersonSchema().load({"name": "David"})) # => {'name': 'David'}
```

Generated schemas are not added to the class registry and therefore cannot be referred to by name in *Nested* fields.

Parameters

- **fields** (*dict*) – Dictionary mapping field names to field instances.
- **name** (*str*) – Optional name for the class, which will appear in the repr for the class.

New in version 3.0.0.

get_attribute(obj: Any, attr: str, default: Any)

Defines how to pull values from an object to serialize.

New in version 2.0.0.

Changed in version 3.0.0a1: Changed position of *obj* and *attr*.

handle_error(error: marshmallow.exceptions.ValidationError, data: Any, *, many: bool, **kwargs)

Custom error handler function for the schema.

Parameters

- **error** – The *ValidationError* raised during (de)serialization.
- **data** – The original input data.
- **many** – Value of *many* on dump or load.
- **partial** – Value of *partial* on load.

New in version 2.0.0.

Changed in version 3.0.0rc9: Receives *many* and *partial* (on deserialization) as keyword arguments.

jsonify(obj, many=<object object>, *args, **kwargs)

Return a JSON response containing the serialized data.

Parameters

- **obj** – Object to serialize.

- **many** (*bool*) – Whether *obj* should be serialized as an instance or as a collection. If unset, defaults to the value of the *many* attribute on this Schema.
- **kwargs** – Additional keyword arguments passed to *flask jsonify*.

Changed in version 0.6.0: Takes the same arguments as *marshmallow.Schema.dump*. Additional keyword arguments are passed to *flask jsonify*.

Changed in version 0.6.3: The *many* argument for this method defaults to the value of the *many* attribute on the Schema. Previously, the *many* argument of this method defaulted to False, regardless of the value of *Schema.many*.

load(*data: Union[Mapping[str, Any], Iterable[Mapping[str, Any]]], *, many: Optional[bool] = None, partial: Optional[Union[bool, Sequence[str], Set[str]]] = None, unknown: Optional[str] = None*)
Deserialize a data structure to an object defined by this Schema's fields.

Parameters

- **data** – The data to deserialize.
- **many** – Whether to deserialize *data* as a collection. If *None*, the value for *self.many* is used.
- **partial** – Whether to ignore missing fields and not require any fields declared. Propagates down to *Nested* fields as well. If its value is an iterable, only missing fields listed in that iterable will be ignored. Use dot delimiters to specify nested fields.
- **unknown** – Whether to exclude, include, or raise an error for unknown fields in the data. Use *EXCLUDE*, *INCLUDE* or *RAISE*. If *None*, the value for *self.unknown* is used.

Returns Deserialized data

New in version 1.0.0.

Changed in version 3.0.0b7: This method returns the deserialized data rather than a (*data*, *errors*) tuple. A *ValidationError* is raised if invalid data are passed.

loads(*json_data: str, *, many: Optional[bool] = None, partial: Optional[Union[bool, Sequence[str], Set[str]]] = None, unknown: Optional[str] = None, **kwargs*)
Same as *load()*, except it takes a JSON string as input.

Parameters

- **json_data** – A JSON string of the data to deserialize.
- **many** – Whether to deserialize *obj* as a collection. If *None*, the value for *self.many* is used.
- **partial** – Whether to ignore missing fields and not require any fields declared. Propagates down to *Nested* fields as well. If its value is an iterable, only missing fields listed in that iterable will be ignored. Use dot delimiters to specify nested fields.
- **unknown** – Whether to exclude, include, or raise an error for unknown fields in the data. Use *EXCLUDE*, *INCLUDE* or *RAISE*. If *None*, the value for *self.unknown* is used.

Returns Deserialized data

New in version 1.0.0.

Changed in version 3.0.0b7: This method returns the deserialized data rather than a (*data*, *errors*) tuple. A *ValidationError* is raised if invalid data are passed.

make_object(*data, **kwargs*)

on_bind_field(*field_name: str, field_obj: marshmallow.fields.Field*) → *None*
Hook to modify a field when it is bound to the *Schema*.

No-op by default.

`opts = <marshmallow.schema.SchemaOpts object>`

property `set_class`: `type`

validate(*data*: Union[Mapping[str, Any], Iterable[Mapping[str, Any]]], *, *many*: Optional[bool] = None, *partial*: Optional[Union[bool, Sequence[str], Set[str]]] = None) → Dict[str, List[str]]

Validate *data* against the schema, returning a dictionary of validation errors.

Parameters

- **data** – The data to validate.
- **many** – Whether to validate *data* as a collection. If *None*, the value for *self.many* is used.
- **partial** – Whether to ignore missing fields and not require any fields declared. Propagates down to *Nested* fields as well. If its value is an iterable, only missing fields listed in that iterable will be ignored. Use dot delimiters to specify nested fields.

Returns A dictionary of validation errors.

New in version 1.1.0.

validate_token(*token*)

```
class app.serializers.auth.AuthUserLoginSerializer(*, only: Optional[Union[Sequence[str], Set[str]]]
                                                    = None, exclude: Union[Sequence[str], Set[str]]
                                                    = (), many: bool = False, context: Optional[Dict]
                                                    = None, load_only: Union[Sequence[str],
Set[str]] = (), dump_only: Union[Sequence[str],
Set[str]] = (), partial: Union[bool, Sequence[str],
Set[str]] = False, unknown: Optional[str] =
None)
```

class Meta

Options object for a Schema.

Example usage:

```
class Meta:
    fields = ("id", "email", "date_created")
    exclude = ("password", "secret_attribute")
```

Available options:

- **fields**: Tuple or list of fields to include in the serialized result.
- **additional**: Tuple or list of fields to include in addition to the explicitly declared fields. **additional** and **fields** are mutually-exclusive options.
- **include**: Dictionary of additional fields to include in the schema. It is usually better to define fields as class variables, but you may need to use this option, e.g., if your fields are Python keywords. May be an *OrderedDict*.
- **exclude**: Tuple or list of fields to exclude in the serialized result. Nested fields can be represented with dot delimiters.
- **dateformat**: Default format for *Date* <fields.Date> fields.
- **datetimeformat**: Default format for *DateTime* <fields.DateTime> fields.
- **timeformat**: Default format for *Time* <fields.Time> fields.

- **render_module:** Module to use for *loads* <Schema.loads> and *dumps* <Schema.dumps>. Defaults to *json* from the standard library.
- **ordered:** If *True*, order serialization output according to the order in which fields were declared. Output of *Schema.dump* will be a *collections.OrderedDict*.
- **index_errors:** If *True*, errors dictionaries will include the *index* of invalid items in a collection.
- **load_only:** Tuple or list of fields to exclude from serialized results.
- **dump_only:** Tuple or list of fields to exclude from deserialization
- **unknown:** Whether to exclude, include, or raise an error for unknown fields in the data. Use *EXCLUDE*, *INCLUDE* or *RAISE*.
- **register:** Whether to register the *Schema* with marshmallow's internal class registry. Must be *True* if you intend to refer to this *Schema* by class name in *Nested* fields. Only set this to *False* when memory usage is critical. Defaults to *True*.

OPTIONS_CLASS

alias of `marshmallow.schema.SchemaOpts`

```
TYPE_MAPPING = {<class 'str':>: <class 'marshmallow.fields.String'>, <class  
'bytes':>: <class 'marshmallow.fields.String'>, <class 'datetime.datetime':>: <class  
'marshmallow.fields.DateTime'>, <class 'float':>: <class  
'marshmallow.fields.Float'>, <class 'bool':>: <class 'marshmallow.fields.Boolean'>,  
<class 'tuple':>: <class 'marshmallow.fields.Raw'>, <class 'list':>: <class  
'marshmallow.fields.Raw'>, <class 'set':>: <class 'marshmallow.fields.Raw'>, <class  
'int':>: <class 'marshmallow.fields.Integer'>, <class 'uuid.UUID':>: <class  
'marshmallow.fields.UUID'>, <class 'datetime.time':>: <class  
'marshmallow.fields.Time'>, <class 'datetime.date':>: <class  
'marshmallow.fields.Date'>, <class 'datetime.timedelta':>: <class  
'marshmallow.fields.TimeDelta'>, <class 'decimal.Decimal':>: <class  
'marshmallow.fields.Decimal'>}
```

`__user` = `None`

`_bind_field(field_name: str, field_obj: marshmallow.fields.Field) → None`

Bind field to the schema, setting any necessary attributes on the field (e.g. parent and name).

Also set field `load_only` and `dump_only` values if `field_name` was specified in class `Meta`.

`static _call_and_store(getter_func, data, *, field_name, error_store, index=None)`

Call `getter_func` with `data` as its argument, and store any *ValidationErrors*.

Parameters

- **getter_func** (*callable*) – Function for getting the serialized/deserialized value from `data`.
- **data** – The data passed to `getter_func`.
- **field_name** (*str*) – Field name.
- **index** (*int*) – Index of the item being validated, if validating a collection, otherwise *None*.


```

_declared_fields = {'email': <fields.String(dump_default=<marshmallow.missing>,
attribute=None, validate=None, required=True, load_only=True, dump_only=False,
load_default=<marshmallow.missing>, allow_none=False, error_messages={'required':
'Missing data for required field.', 'null': 'Field may not be null.',
'validator_failed': 'Invalid value.', 'invalid': 'Not a valid string.',
'invalid_utf8': 'Not a valid utf-8 string.'})>, 'password':
<fields.String(dump_default=<marshmallow.missing>, attribute=None,
validate=<Length(min=8, max=50, equal=None, error=None)>, required=True,
load_only=True, dump_only=False, load_default=<marshmallow.missing>,
allow_none=False, error_messages={'required': 'Missing data for required field.',
'null': 'Field may not be null.', 'validator_failed': 'Invalid value.', 'invalid':
'Not a valid string.', 'invalid_utf8': 'Not a valid utf-8 string.'})>}>
_default_error_messages = {'type': 'Invalid input type.', 'unknown': 'Unknown
field.'}

_deserialize(data: Union[Mapping[str, Any], Iterable[Mapping[str, Any]]], *, error_store:
marshmallow.error_store.ErrorStore, many: bool = False, partial=False, unknown='raise',
index=None) → Union[marshmallow.schema._T, List[marshmallow.schema._T]]

```

Deserialize data.

Parameters

- **data** (*dict*) – The data to deserialize.
- **error_store** (*ErrorStore*) – Structure to store errors.
- **many** (*bool*) – *True* if data should be deserialized as a collection.
- **partial** (*bool | tuple*) – Whether to ignore missing fields and not require any fields declared. Propagates down to Nested fields as well. If its value is an iterable, only missing fields listed in that iterable will be ignored. Use dot delimiters to specify nested fields.
- **unknown** – Whether to exclude, include, or raise an error for unknown fields in the data. Use *EXCLUDE*, *INCLUDE* or *RAISE*.
- **index** (*int*) – Index of the item being serialized (for storing errors) if serializing a collection, otherwise *None*.

Returns A dictionary of the deserialized data.

```

_do_load(data: Union[Mapping[str, Any], Iterable[Mapping[str, Any]]], *, many: Optional[bool] = None,
partial: Optional[Union[bool, Sequence[str], Set[str]]] = None, unknown: Optional[str] = None,
postprocess: bool = True)

```

Deserialize *data*, returning the deserialized result. This method is private API.

Parameters

- **data** – The data to deserialize.
- **many** – Whether to deserialize *data* as a collection. If *None*, the value for *self.many* is used.
- **partial** – Whether to validate required fields. If its value is an iterable, only fields listed in that iterable will be ignored will be allowed missing. If *True*, all fields will be allowed missing. If *None*, the value for *self.partial* is used.
- **unknown** – Whether to exclude, include, or raise an error for unknown fields in the data. Use *EXCLUDE*, *INCLUDE* or *RAISE*. If *None*, the value for *self.unknown* is used.
- **postprocess** – Whether to run *post_load* methods..

Returns Deserialized data

```
_has_processors(tag) → bool

_hooks = defaultdict(<class 'list'>, {'post_load', False): ['make_object'],
'validates': ['validate_email', 'validate_password']})

_init_fields() → None
    Update self.fields, self.load_fields, and self.dump_fields based on schema options. This method is private
    API.

_invoke_dump_processors(tag: str, data, *, many: bool, original_data=None)

_invoke_field_validators(*, error_store: marshmallow.error_store.ErrorStore, data, many: bool)

_invoke_load_processors(tag: str, data, *, many: bool, original_data, partial: Union[bool,
    Sequence[str], Set[str]])

_invoke_processors(tag: str, *, pass_many: bool, data, many: bool, original_data=None, **kwargs)

_invoke_schema_validators(*, error_store: marshmallow.error_store.ErrorStore, pass_many: bool, data,
    original_data, many: bool, partial: Union[bool, Sequence[str], Set[str]],
    field_errors: bool = False)

_normalize_nested_options() → None
    Apply then flatten nested schema options. This method is private API.

_run_validator(validator_func, output, *, original_data, error_store, many, partial, pass_original,
    index=None)

_serialize(obj: Union[marshmallow.schema._T, Iterable[marshmallow.schema._T]], *, many: bool =
    False)
    Serialize obj.
```

Parameters

- **obj** – The object(s) to serialize.
- **many** (*bool*) – *True* if data should be serialized as a collection.

Returns A dictionary of the serialized data

Changed in version 1.0.0: Renamed from `marshal`.

property dict_class: **type**

```
dump(obj: Any, *, many: Optional[bool] = None)
```

Serialize an object to native Python data types according to this Schema's fields.

Parameters

- **obj** – The object to serialize.
- **many** – Whether to serialize *obj* as a collection. If *None*, the value for *self.many* is used.

Returns Serialized data

New in version 1.0.0.

Changed in version 3.0.0b7: This method returns the serialized data rather than a (data, errors) tuple. A `ValidationError` is raised if *obj* is invalid.

Changed in version 3.0.0rc9: Validation no longer occurs upon serialization.

```
dumps(obj: Any, *args, many: Optional[bool] = None, **kwargs)
```

Same as `dump()`, except return a JSON-encoded string.

Parameters

- **obj** – The object to serialize.
- **many** – Whether to serialize *obj* as a collection. If *None*, the value for *self.many* is used.

Returns A json string

New in version 1.0.0.

Changed in version 3.0.0b7: This method returns the serialized data rather than a (data, errors) tuple. A *ValidationError* is raised if *obj* is invalid.

error_messages = {}

Overrides for default schema-level error messages

fields

Dictionary mapping field_names -> Field objects

classmethod from_dict(fields: Dict[str, Union[marshmallow.fields.Field, type]], *, name: str = 'GeneratedSchema') → type

Generate a *Schema* class given a dictionary of fields.

```
from marshmallow import Schema, fields

PersonSchema = Schema.from_dict({"name": fields.Str()})
print(PersonSchema().load({"name": "David"})) # => {'name': 'David'}
```

Generated schemas are not added to the class registry and therefore cannot be referred to by name in *Nested* fields.

Parameters

- **fields** (*dict*) – Dictionary mapping field names to field instances.
- **name** (*str*) – Optional name for the class, which will appear in the repr for the class.

New in version 3.0.0.

get_attribute(obj: Any, attr: str, default: Any)

Defines how to pull values from an object to serialize.

New in version 2.0.0.

Changed in version 3.0.0a1: Changed position of *obj* and *attr*.

handle_error(error: marshmallow.exceptions.ValidationError, data: Any, *, many: bool, **kwargs)

Custom error handler function for the schema.

Parameters

- **error** – The *ValidationError* raised during (de)serialization.
- **data** – The original input data.
- **many** – Value of *many* on dump or load.
- **partial** – Value of *partial* on load.

New in version 2.0.0.

Changed in version 3.0.0rc9: Receives *many* and *partial* (on deserialization) as keyword arguments.

jsonify(obj, many=<object object>, *args, **kwargs)

Return a JSON response containing the serialized data.

Parameters

- **obj** – Object to serialize.

- **many** (*bool*) – Whether *obj* should be serialized as an instance or as a collection. If unset, defaults to the value of the *many* attribute on this Schema.
- **kwargs** – Additional keyword arguments passed to *flask jsonify*.

Changed in version 0.6.0: Takes the same arguments as *marshmallow.Schema.dump*. Additional keyword arguments are passed to *flask jsonify*.

Changed in version 0.6.3: The *many* argument for this method defaults to the value of the *many* attribute on the Schema. Previously, the *many* argument of this method defaulted to False, regardless of the value of *Schema.many*.

load(*data: Union[Mapping[str, Any], Iterable[Mapping[str, Any]]], *, many: Optional[bool] = None, partial: Optional[Union[bool, Sequence[str], Set[str]]] = None, unknown: Optional[str] = None*)
Deserialize a data structure to an object defined by this Schema's fields.

Parameters

- **data** – The data to deserialize.
- **many** – Whether to deserialize *data* as a collection. If *None*, the value for *self.many* is used.
- **partial** – Whether to ignore missing fields and not require any fields declared. Propagates down to *Nested* fields as well. If its value is an iterable, only missing fields listed in that iterable will be ignored. Use dot delimiters to specify nested fields.
- **unknown** – Whether to exclude, include, or raise an error for unknown fields in the data. Use *EXCLUDE*, *INCLUDE* or *RAISE*. If *None*, the value for *self.unknown* is used.

Returns Deserialized data

New in version 1.0.0.

Changed in version 3.0.0b7: This method returns the deserialized data rather than a (*data*, *errors*) tuple. A *ValidationError* is raised if invalid data are passed.

loads(*json_data: str, *, many: Optional[bool] = None, partial: Optional[Union[bool, Sequence[str], Set[str]]] = None, unknown: Optional[str] = None, **kwargs*)
Same as *load()*, except it takes a JSON string as input.

Parameters

- **json_data** – A JSON string of the data to deserialize.
- **many** – Whether to deserialize *obj* as a collection. If *None*, the value for *self.many* is used.
- **partial** – Whether to ignore missing fields and not require any fields declared. Propagates down to *Nested* fields as well. If its value is an iterable, only missing fields listed in that iterable will be ignored. Use dot delimiters to specify nested fields.
- **unknown** – Whether to exclude, include, or raise an error for unknown fields in the data. Use *EXCLUDE*, *INCLUDE* or *RAISE*. If *None*, the value for *self.unknown* is used.

Returns Deserialized data

New in version 1.0.0.

Changed in version 3.0.0b7: This method returns the deserialized data rather than a (*data*, *errors*) tuple. A *ValidationError* is raised if invalid data are passed.

make_object(*data, **kwargs*)

on_bind_field(*field_name: str, field_obj: marshmallow.fields.Field*) → None
Hook to modify a field when it is bound to the *Schema*.

No-op by default.

`opts = <marshmallow.schema.SchemaOpts object>`

property `set_class`: `type`

validate(*data*: Union[Mapping[str, Any], Iterable[Mapping[str, Any]]], *, *many*: Optional[bool] = None, *partial*: Optional[Union[bool, Sequence[str], Set[str]]] = None) → Dict[str, List[str]]

Validate *data* against the schema, returning a dictionary of validation errors.

Parameters

- **data** – The data to validate.
- **many** – Whether to validate *data* as a collection. If *None*, the value for *self.many* is used.
- **partial** – Whether to ignore missing fields and not require any fields declared. Propagates down to *Nested* fields as well. If its value is an iterable, only missing fields listed in that iterable will be ignored. Use dot delimiters to specify nested fields.

Returns A dictionary of validation errors.

New in version 1.1.0.

validate_email(*email*)

validate_password(*password*)

app.serializers.core

Description

Classes

SearchSerializer(*[, only, exclude, many, ...])

TimestampField(*, load_default, missing, ...)

Field that serializes to timestamp integer and deserializes to a datetime.datetime class.

app.serializers.core.SearchSerializer

```
class app.serializers.core.SearchSerializer(*, only: Optional[Union[Sequence[str], Set[str]]] = None,
    exclude: Union[Sequence[str], Set[str]] = (), many: bool = False, context: Optional[Dict] = None, load_only: Union[Sequence[str], Set[str]] = (), dump_only: Union[Sequence[str], Set[str]] = (), partial: Union[bool, Sequence[str], Set[str]] = False, unknown: Optional[str] = None)
```

Bases: flask_marshmallow.schema.Schema

Attributes

<i>SearchSerializer.TYPE_MAPPING</i>	
<i>SearchSerializer.dict_class</i>	
<i>SearchSerializer.error_messages</i>	Overrides for default schema-level error messages
<i>SearchSerializer.opts</i>	
<i>SearchSerializer.set_class</i>	

app.serializers.core.SearchSerializer.TYPE_MAPPING

```
SearchSerializer.TYPE_MAPPING = {<class 'str'>: <class  
'marshmallow.fields.String'>, <class 'bytes'>: <class  
'marshmallow.fields.String'>, <class 'datetime.datetime'>: <class  
'marshmallow.fields.DateTime'>, <class 'float'>: <class  
'marshmallow.fields.Float'>, <class 'bool'>: <class  
'marshmallow.fields.Boolean'>, <class 'tuple'>: <class  
'marshmallow.fields.Raw'>, <class 'list'>: <class  
'marshmallow.fields.Raw'>, <class 'set'>: <class 'marshmallow.fields.Raw'>,  
<class 'int'>: <class 'marshmallow.fields.Integer'>, <class 'uuid.UUID'>:  
<class 'marshmallow.fields.UUID'>, <class 'datetime.time'>: <class  
'marshmallow.fields.Time'>, <class 'datetime.date'>: <class  
'marshmallow.fields.Date'>, <class 'datetime.timedelta'>: <class  
'marshmallow.fields.TimeDelta'>, <class 'decimal.Decimal'>: <class  
'marshmallow.fields.Decimal'>}
```

app.serializers.core.SearchSerializer.dict_class

property SearchSerializer.dict_class: type

app.serializers.core.SearchSerializer.error_messages

SearchSerializer.error_messages = {}
Overrides for default schema-level error messages

app.serializers.core.SearchSerializer.opts

SearchSerializer.opts = <marshmallow.schema.SchemaOpts object>

app.serializers.core.SearchSerializer.set_class

property SearchSerializer.set_class: type

Methods

<code>SearchSerializer.__init__(*[, only, ...])</code>	
<code>SearchSerializer.dump(obj, *[, many])</code>	Serialize an object to native Python data types according to this Schema's fields.
<code>SearchSerializer.dumps(obj, *args[, many])</code>	Same as <code>dump()</code> , except return a JSON-encoded string.
<code>SearchSerializer.from_dict(fields, *[, name])</code>	Generate a <i>Schema</i> class given a dictionary of fields.
<code>SearchSerializer.get_attribute(obj, attr, ...)</code>	Defines how to pull values from an object to serialize.
<code>SearchSerializer.handle_error(error, data, ...)</code>	Custom error handler function for the schema.
<code>SearchSerializer.jsonify(obj[, many])</code>	Return a JSON response containing the serialized data.
<code>SearchSerializer.load(data, *[, many, ...])</code>	Deserialize a data structure to an object defined by this Schema's fields.
<code>SearchSerializer.loads(json_data, *[, many, ...])</code>	Same as <code>load()</code> , except it takes a JSON string as input.
<code>SearchSerializer.on_bind_field(field_name, ...)</code>	Hook to modify a field when it is bound to the <i>Schema</i> .
<code>SearchSerializer.validate(data, *[, many, ...])</code>	Validate <i>data</i> against the schema, returning a dictionary of validation errors.

app.serializers.core.SearchSerializer.__init__

`SearchSerializer.__init__(*, only: Optional[Union[Sequence[str], Set[str]]] = None, exclude: Union[Sequence[str], Set[str]] = (), many: bool = False, context: Optional[Dict] = None, load_only: Union[Sequence[str], Set[str]] = (), dump_only: Union[Sequence[str], Set[str]] = (), partial: Union[bool, Sequence[str], Set[str]] = False, unknown: Optional[str] = None)`

app.serializers.core.SearchSerializer.dump

`SearchSerializer.dump(obj: Any, *, many: Optional[bool] = None)`

Serialize an object to native Python data types according to this Schema's fields.

Parameters

- **obj** – The object to serialize.
- **many** – Whether to serialize *obj* as a collection. If *None*, the value for *self.many* is used.

Returns Serialized data

New in version 1.0.0.

Changed in version 3.0.0b7: This method returns the serialized data rather than a (data, errors) tuple. A `ValidationError` is raised if `obj` is invalid.

Changed in version 3.0.0rc9: Validation no longer occurs upon serialization.

`app.serializers.core.SearchSerializer.dumps`

`SearchSerializer.dumps(obj: Any, *args, many: Optional[bool] = None, **kwargs)`

Same as `dump()`, except return a JSON-encoded string.

Parameters

- **obj** – The object to serialize.
- **many** – Whether to serialize *obj* as a collection. If *None*, the value for *self.many* is used.

Returns A json string

New in version 1.0.0.

Changed in version 3.0.0b7: This method returns the serialized data rather than a (data, errors) tuple. A `ValidationError` is raised if `obj` is invalid.

`app.serializers.core.SearchSerializer.from_dict`

classmethod `SearchSerializer.from_dict(fields: Dict[str, Union[marshmallow.fields.Field, type]], *, name: str = 'GeneratedSchema') → type`

Generate a *Schema* class given a dictionary of fields.

```
from marshmallow import Schema, fields

PersonSchema = Schema.from_dict({"name": fields.Str()})
print(PersonSchema().load({"name": "David"})) # => {'name': 'David'}
```

Generated schemas are not added to the class registry and therefore cannot be referred to by name in *Nested* fields.

Parameters

- **fields** (*dict*) – Dictionary mapping field names to field instances.
- **name** (*str*) – Optional name for the class, which will appear in the `repr` for the class.

New in version 3.0.0.

`app.serializers.core.SearchSerializer.get_attribute`

`SearchSerializer.get_attribute(obj: Any, attr: str, default: Any)`

Defines how to pull values from an object to serialize.

New in version 2.0.0.

Changed in version 3.0.0a1: Changed position of `obj` and `attr`.

app.serializers.core.SearchSerializer.handle_error

`SearchSerializer.handle_error`(*error: marshmallow.exceptions.ValidationError*, *data: Any*,
*, *many: bool*, ***kwargs*)

Custom error handler function for the schema.

Parameters

- **error** – The *ValidationError* raised during (de)serialization.
- **data** – The original input data.
- **many** – Value of *many* on dump or load.
- **partial** – Value of *partial* on load.

New in version 2.0.0.

Changed in version 3.0.0rc9: Receives *many* and *partial* (on deserialization) as keyword arguments.

app.serializers.core.SearchSerializer.jsonify

`SearchSerializer.jsonify`(*obj*, *many=<object object>*, **args*, ***kwargs*)

Return a JSON response containing the serialized data.

Parameters

- **obj** – Object to serialize.
- **many** (*bool*) – Whether *obj* should be serialized as an instance or as a collection. If unset, defaults to the value of the *many* attribute on this Schema.
- **kwargs** – Additional keyword arguments passed to *flask.jsonify*.

Changed in version 0.6.0: Takes the same arguments as *marshmallow.Schema.dump*. Additional keyword arguments are passed to *flask.jsonify*.

Changed in version 0.6.3: The *many* argument for this method defaults to the value of the *many* attribute on the Schema. Previously, the *many* argument of this method defaulted to *False*, regardless of the value of *Schema.many*.

app.serializers.core.SearchSerializer.load

`SearchSerializer.load`(*data: Union[Mapping[str, Any], Iterable[Mapping[str, Any]]]*, *,
many: Optional[bool] = None, *partial: Optional[Union[bool, Sequence[str], Set[str]]] = None*, *unknown: Optional[str] = None*)

Deserialize a data structure to an object defined by this Schema's fields.

Parameters

- **data** – The data to deserialize.
- **many** – Whether to deserialize *data* as a collection. If *None*, the value for *self.many* is used.
- **partial** – Whether to ignore missing fields and not require any fields declared. Propagates down to *Nested* fields as well. If its value is an iterable, only missing fields listed in that iterable will be ignored. Use dot delimiters to specify nested fields.
- **unknown** – Whether to exclude, include, or raise an error for unknown fields in the data. Use *EXCLUDE*, *INCLUDE* or *RAISE*. If *None*, the value for *self.unknown* is used.

Returns Deserialized data

New in version 1.0.0.

Changed in version 3.0.0b7: This method returns the deserialized data rather than a (*data*, *errors*) tuple. A *ValidationError* is raised if invalid data are passed.

app.serializers.core.SearchSerializer.loads

`SearchSerializer.loads(json_data: str, *, many: Optional[bool] = None, partial: Optional[Union[bool, Sequence[str], Set[str]]] = None, unknown: Optional[str] = None, **kwargs)`

Same as `load()`, except it takes a JSON string as input.

Parameters

- **json_data** – A JSON string of the data to deserialize.
- **many** – Whether to deserialize *obj* as a collection. If *None*, the value for *self.many* is used.
- **partial** – Whether to ignore missing fields and not require any fields declared. Propagates down to *Nested* fields as well. If its value is an iterable, only missing fields listed in that iterable will be ignored. Use dot delimiters to specify nested fields.
- **unknown** – Whether to exclude, include, or raise an error for unknown fields in the data. Use *EXCLUDE*, *INCLUDE* or *RAISE*. If *None*, the value for *self.unknown* is used.

Returns Deserialized data

New in version 1.0.0.

Changed in version 3.0.0b7: This method returns the deserialized data rather than a (data, errors) tuple. A `ValidationError` is raised if invalid data are passed.

app.serializers.core.SearchSerializer.on_bind_field

`SearchSerializer.on_bind_field(field_name: str, field_obj: marshmallow.fields.Field) → None`

Hook to modify a field when it is bound to the *Schema*.

No-op by default.

app.serializers.core.SearchSerializer.validate

`SearchSerializer.validate(data: Union[Mapping[str, Any], Iterable[Mapping[str, Any]]], *, many: Optional[bool] = None, partial: Optional[Union[bool, Sequence[str], Set[str]]] = None) → Dict[str, List[str]]`

Validate *data* against the schema, returning a dictionary of validation errors.

Parameters

- **data** – The data to validate.
- **many** – Whether to validate *data* as a collection. If *None*, the value for *self.many* is used.
- **partial** – Whether to ignore missing fields and not require any fields declared. Propagates down to *Nested* fields as well. If its value is an iterable, only missing fields listed in that iterable will be ignored. Use dot delimiters to specify nested fields.

Returns A dictionary of validation errors.

New in version 1.1.0.

app.serializers.core.TimestampField

```
class app.serializers.core.TimestampField(*, load_default: Any = <marshmallow.missing>, missing:
    Any = <marshmallow.missing>, dump_default: Any =
    <marshmallow.missing>, default: Any =
    <marshmallow.missing>, data_key: Optional[str] = None,
    attribute: Optional[str] = None, validate:
    Optional[Union[Callable[[Any], Any],
    Iterable[Callable[[Any], Any]]]] = None, required: bool =
    False, allow_none: Optional[bool] = None, load_only: bool
    = False, dump_only: bool = False, error_messages:
    Optional[Dict[str, str]] = None, metadata:
    Optional[Mapping[str, Any]] = None,
    **additional_metadata)
```

Bases: `marshmallow.fields.Field`

Field that serializes to timestamp integer and deserializes to a `datetime.datetime` class.

Attributes

<code>TimestampField.context</code>	The context dictionary for the parent Schema.
<code>TimestampField.default</code>	
<code>TimestampField.default_error_messages</code>	Default error messages for various kinds of errors.
<code>TimestampField.missing</code>	
<code>TimestampField.name</code>	
<code>TimestampField.parent</code>	
<code>TimestampField.root</code>	

app.serializers.core.TimestampField.context**property** `TimestampField.context`

The context dictionary for the parent Schema.

app.serializers.core.TimestampField.default

property TimestampField.default

app.serializers.core.TimestampField.default_error_messages

```
TimestampField.default_error_messages = {'null': 'Field may not be null.',
'required': 'Missing data for required field.', 'validator_failed':
'Invalid value.'}
```

Default error messages for various kinds of errors. The keys in this dictionary are passed to *Field.make_error*. The values are error messages passed to *marshmallow.exceptions.ValidationError*.

app.serializers.core.TimestampField.missing

property TimestampField.missing

app.serializers.core.TimestampField.name

TimestampField.name = None

app.serializers.core.TimestampField.parent

TimestampField.parent = None

app.serializers.core.TimestampField.root

TimestampField.root = None

Methods

<code>TimestampField.__init__(*[, load_default, ...])</code>	
<code>TimestampField.deserialize(value[, attr, data])</code>	Deserialize value.
<code>TimestampField.fail(key, **kwargs)</code>	Helper method that raises a <i>ValidationError</i> with an error message from <i>self.error_messages</i> .
<code>TimestampField.get_value(obj, attr[, ...])</code>	Return the value for a given key from an object.
<code>TimestampField.make_error(key, **kwargs)</code>	Helper method to make a <i>ValidationError</i> with an error message from <i>self.error_messages</i> .
<code>TimestampField.serialize(attr, obj[, accessor])</code>	Pulls the value for the given key from the object, applies the field's formatting and returns the result.

app.serializers.core.TimestampField.__init__

```
TimestampField.__init__(*, load_default: Any = <marshmallow.missing>, missing: Any =
    <marshmallow.missing>, dump_default: Any =
    <marshmallow.missing>, default: Any = <marshmallow.missing>,
    data_key: Optional[str] = None, attribute: Optional[str] = None,
    validate: Optional[Union[Callable[[Any], Any],
    Iterable[Callable[[Any], Any]]]] = None, required: bool = False,
    allow_none: Optional[bool] = None, load_only: bool = False,
    dump_only: bool = False, error_messages: Optional[Dict[str, str]]
    = None, metadata: Optional[Mapping[str, Any]] = None,
    **additional_metadata) → None
```

app.serializers.core.TimestampField.deserialize

```
TimestampField.deserialize(value: Any, attr: Optional[str] = None, data:
    Optional[Mapping[str, Any]] = None, **kwargs)
```

Deserialize value.

Parameters

- **value** – The value to deserialize.
- **attr** – The attribute/key in *data* to deserialize.
- **data** – The raw input data passed to *Schema.load*.
- **kwargs** – Field-specific keyword arguments.

Raises `ValidationError` – If an invalid value is passed or if a required value is missing.

app.serializers.core.TimestampField.fail

```
TimestampField.fail(key: str, **kwargs)
```

Helper method that raises a *ValidationError* with an error message from *self.error_messages*.

Deprecated since version 3.0.0: Use *make_error* <marshmallow.fields.Field.make_error> instead.

app.serializers.core.TimestampField.get_value

```
TimestampField.get_value(obj, attr, accessor=None, default=<marshmallow.missing>)
```

Return the value for a given key from an object.

Parameters

- **obj** (*object*) – The object to get the value from.
- **attr** (*str*) – The attribute/key in *obj* to get the value from.
- **accessor** (*callable*) – A callable used to retrieve the value of *attr* from the object *obj*. Defaults to *marshmallow.utils.get_value*.

app.serializers.core.TimestampField.make_error

`TimestampField.make_error(key: str, **kwargs) → marshmallow.exceptions.ValidationError`
Helper method to make a *ValidationError* with an error message from `self.error_messages`.

app.serializers.core.TimestampField.serialize

`TimestampField.serialize(attr: str, obj: Any, accessor: Optional[Callable[[Any, str], Any], Any]] = None, **kwargs)`

Pulls the value for the given key from the object, applies the field's formatting and returns the result.

Parameters

- **attr** – The attribute/key to get from the object.
- **obj** – The object to access the attribute/key from.
- **accessor** – Function used to access values from `obj`.
- **kwargs** – Field-specific keyword arguments.

```
class app.serializers.core.SearchSerializer(*, only: Optional[Union[Sequence[str], Set[str]]] = None,
                                           exclude: Union[Sequence[str], Set[str]] = (), many: bool =
                                           False, context: Optional[Dict] = None, load_only:
                                           Union[Sequence[str], Set[str]] = (), dump_only:
                                           Union[Sequence[str], Set[str]] = (), partial: Union[bool,
                                           Sequence[str], Set[str]] = False, unknown: Optional[str] =
                                           None)
```

class Meta

Options object for a Schema.

Example usage:

```
class Meta:
    fields = ("id", "email", "date_created")
    exclude = ("password", "secret_attribute")
```

Available options:

- **fields**: Tuple or list of fields to include in the serialized result.
- **additional**: Tuple or list of fields to include *in addition to the* explicitly declared fields. **additional** and **fields** are mutually-exclusive options.
- **include**: Dictionary of additional fields to include in the schema. It is usually better to define fields as class variables, but you may need to use this option, e.g., if your fields are Python keywords. May be an *OrderedDict*.
- **exclude**: Tuple or list of fields to exclude in the serialized result. Nested fields can be represented with dot delimiters.
- **dateformat**: Default format for *Date* <fields.Date> fields.
- **datetimeformat**: Default format for *DateTime* <fields.DateTime> fields.
- **timeformat**: Default format for *Time* <fields.Time> fields.
- **render_module**: Module to use for *loads* <Schema.loads> and *dumps* <Schema.dumps>. Defaults to *json* from the standard library.

- **ordered:** If *True*, order serialization output according to the order in which fields were declared. Output of *Schema.dump* will be a *collections.OrderedDict*.
- **index_errors:** If *True*, errors dictionaries will include the index of invalid items in a collection.
- **load_only:** Tuple or list of fields to exclude from serialized results.
- **dump_only:** Tuple or list of fields to exclude from deserialization
- **unknown:** Whether to exclude, include, or raise an error for unknown fields in the data. Use *EXCLUDE*, *INCLUDE* or *RAISE*.
- **register:** Whether to register the *Schema* with marshmallow's internal class registry. Must be *True* if you intend to refer to this *Schema* by class name in *Nested* fields. Only set this to *False* when memory usage is critical. Defaults to *True*.

OPTIONS_CLASS

alias of `marshmallow.schema.SchemaOpts`

```
TYPE_MAPPING = {<class 'str':>: <class 'marshmallow.fields.String'>, <class
'bytes':>: <class 'marshmallow.fields.String'>, <class 'datetime.datetime':>: <class
'marshmallow.fields.DateTimeField'>, <class 'float':>: <class
'marshmallow.fields.Float'>, <class 'bool':>: <class 'marshmallow.fields.Boolean'>,
<class 'tuple':>: <class 'marshmallow.fields.Raw'>, <class 'list':>: <class
'marshmallow.fields.Raw'>, <class 'set':>: <class 'marshmallow.fields.Raw'>, <class
'int':>: <class 'marshmallow.fields.Integer'>, <class 'uuid.UUID':>: <class
'marshmallow.fields.UUID'>, <class 'datetime.time':>: <class
'marshmallow.fields.Time'>, <class 'datetime.date':>: <class
'marshmallow.fields.Date'>, <class 'datetime.timedelta':>: <class
'marshmallow.fields.TimeDelta'>, <class 'decimal.Decimal':>: <class
'marshmallow.fields.Decimal'>}
```

_bind_field(*field_name*: str, *field_obj*: marshmallow.fields.Field) → None

Bind field to the schema, setting any necessary attributes on the field (e.g. parent and name).

Also set field *load_only* and *dump_only* values if *field_name* was specified in class *Meta*.

static _call_and_store(*getter_func*, *data*, *, *field_name*, *error_store*, *index=None*)

Call *getter_func* with *data* as its argument, and store any *ValidationErrors*.

Parameters

- **getter_func** (*callable*) – Function for getting the serialized/deserialized value from *data*.
- **data** – The data passed to *getter_func*.
- **field_name** (*str*) – Field name.
- **index** (*int*) – Index of the item being validated, if validating a collection, otherwise *None*.

```

_declared_fields = {'items_per_page':
<fields.Integer(dump_default=<marshmallow.missing>, attribute=None, validate=None,
required=False, load_only=False, dump_only=False,
load_default=<marshmallow.missing>, allow_none=False, error_messages={'required':
'Missing data for required field.', 'null': 'Field may not be null.',
'validator_failed': 'Invalid value.', 'invalid': 'Not a valid integer.',
'too_large': 'Number too large.'})>, 'order':
<fields.List(dump_default=<marshmallow.missing>, attribute=None, validate=None,
required=False, load_only=False, dump_only=False,
load_default=<marshmallow.missing>, allow_none=False, error_messages={'required':
'Missing data for required field.', 'null': 'Field may not be null.',
'validator_failed': 'Invalid value.', 'invalid': 'Not a valid list.'})>,
'page_number': <fields.Integer(dump_default=<marshmallow.missing>, attribute=None,
validate=None, required=False, load_only=False, dump_only=False,
load_default=<marshmallow.missing>, allow_none=False, error_messages={'required':
'Missing data for required field.', 'null': 'Field may not be null.',
'validator_failed': 'Invalid value.', 'invalid': 'Not a valid integer.',
'too_large': 'Number too large.'})>, 'search':
<fields.List(dump_default=<marshmallow.missing>, attribute=None, validate=None,
required=False, load_only=False, dump_only=False,
load_default=<marshmallow.missing>, allow_none=False, error_messages={'required':
'Missing data for required field.', 'null': 'Field may not be null.',
'validator_failed': 'Invalid value.', 'invalid': 'Not a valid list.'})>}>
_default_error_messages = {'type': 'Invalid input type.', 'unknown': 'Unknown
field.'}

```

```

_deserialize(data: Union[Mapping[str, Any], Iterable[Mapping[str, Any]]], *, error_store:
marshmallow.error_store.ErrorStore, many: bool = False, partial=False, unknown='raise',
index=None) → Union[marshmallow.schema._T, List[marshmallow.schema._T]]

```

Deserialize data.

Parameters

- **data** (*dict*) – The data to deserialize.
- **error_store** (*ErrorStore*) – Structure to store errors.
- **many** (*bool*) – *True* if data should be deserialized as a collection.
- **partial** (*bool | tuple*) – Whether to ignore missing fields and not require any fields declared. Propagates down to Nested fields as well. If its value is an iterable, only missing fields listed in that iterable will be ignored. Use dot delimiters to specify nested fields.
- **unknown** – Whether to exclude, include, or raise an error for unknown fields in the data. Use *EXCLUDE*, *INCLUDE* or *RAISE*.
- **index** (*int*) – Index of the item being serialized (for storing errors) if serializing a collection, otherwise *None*.

Returns A dictionary of the deserialized data.

```

_do_load(data: Union[Mapping[str, Any], Iterable[Mapping[str, Any]]], *, many: Optional[bool] = None,
partial: Optional[Union[bool, Sequence[str], Set[str]]] = None, unknown: Optional[str] = None,
postprocess: bool = True)

```

Deserialize *data*, returning the deserialized result. This method is private API.

Parameters

- **data** – The data to deserialize.

- **many** – Whether to deserialize *data* as a collection. If *None*, the value for *self.many* is used.
- **partial** – Whether to validate required fields. If its value is an iterable, only fields listed in that iterable will be ignored will be allowed missing. If *True*, all fields will be allowed missing. If *None*, the value for *self.partial* is used.
- **unknown** – Whether to exclude, include, or raise an error for unknown fields in the data. Use *EXCLUDE*, *INCLUDE* or *RAISE*. If *None*, the value for *self.unknown* is used.
- **postprocess** – Whether to run *post_load* methods..

Returns Deserialized data

_has_processors(*tag*) → bool

_hooks = {}

_init_fields() → None

Update *self.fields*, *self.load_fields*, and *self.dump_fields* based on schema options. This method is private API.

_invoke_dump_processors(*tag: str, data, *, many: bool, original_data=None*)

_invoke_field_validators(**, error_store: marshmallow.error_store.ErrorStore, data, many: bool*)

_invoke_load_processors(*tag: str, data, *, many: bool, original_data, partial: Union[bool, Sequence[str], Set[str]]*)

_invoke_processors(*tag: str, *, pass_many: bool, data, many: bool, original_data=None, **kwargs*)

_invoke_schema_validators(**, error_store: marshmallow.error_store.ErrorStore, pass_many: bool, data, original_data, many: bool, partial: Union[bool, Sequence[str], Set[str]], field_errors: bool = False*)

_normalize_nested_options() → None

Apply then flatten nested schema options. This method is private API.

_run_validator(*validator_func, output, *, original_data, error_store, many, partial, pass_original, index=None*)

_serialize(*obj: Union[marshmallow.schema._T, Iterable[marshmallow.schema._T]], *, many: bool = False*)

Serialize *obj*.

Parameters

- **obj** – The object(s) to serialize.
- **many** (*bool*) – *True* if *data* should be serialized as a collection.

Returns A dictionary of the serialized data

Changed in version 1.0.0: Renamed from *marshal*.

property dict_class: type

dump(*obj: Any, *, many: Optional[bool] = None*)

Serialize an object to native Python data types according to this Schema's fields.

Parameters

- **obj** – The object to serialize.
- **many** – Whether to serialize *obj* as a collection. If *None*, the value for *self.many* is used.

Returns Serialized data

New in version 1.0.0.

Changed in version 3.0.0b7: This method returns the serialized data rather than a (data, errors) tuple. A `ValidationError` is raised if `obj` is invalid.

Changed in version 3.0.0rc9: Validation no longer occurs upon serialization.

dumps(*obj: Any, *args, many: Optional[bool] = None, **kwargs*)

Same as `dump()`, except return a JSON-encoded string.

Parameters

- **obj** – The object to serialize.
- **many** – Whether to serialize *obj* as a collection. If *None*, the value for *self.many* is used.

Returns A json string

New in version 1.0.0.

Changed in version 3.0.0b7: This method returns the serialized data rather than a (data, errors) tuple. A `ValidationError` is raised if `obj` is invalid.

error_messages = {}

Overrides for default schema-level error messages

fields

Dictionary mapping field_names -> Field objects

classmethod from_dict(*fields: Dict[str, Union[marshmallow.fields.Field, type]], *, name: str = 'GeneratedSchema'*) → type

Generate a *Schema* class given a dictionary of fields.

```
from marshmallow import Schema, fields

PersonSchema = Schema.from_dict({"name": fields.Str()})
print(PersonSchema().load({"name": "David"})) # => {'name': 'David'}
```

Generated schemas are not added to the class registry and therefore cannot be referred to by name in *Nested* fields.

Parameters

- **fields** (*dict*) – Dictionary mapping field names to field instances.
- **name** (*str*) – Optional name for the class, which will appear in the `repr` for the class.

New in version 3.0.0.

get_attribute(*obj: Any, attr: str, default: Any*)

Defines how to pull values from an object to serialize.

New in version 2.0.0.

Changed in version 3.0.0a1: Changed position of `obj` and `attr`.

handle_error(*error: marshmallow.exceptions.ValidationError, data: Any, *, many: bool, **kwargs*)

Custom error handler function for the schema.

Parameters

- **error** – The *ValidationError* raised during (de)serialization.
- **data** – The original input data.
- **many** – Value of `many` on `dump` or `load`.

- **partial** – Value of **partial** on load.

New in version 2.0.0.

Changed in version 3.0.0rc9: Receives *many* and *partial* (on deserialization) as keyword arguments.

jsonify(*obj*, *many*=<object object>, **args*, ***kwargs*)

Return a JSON response containing the serialized data.

Parameters

- **obj** – Object to serialize.
- **many** (*bool*) – Whether *obj* should be serialized as an instance or as a collection. If unset, defaults to the value of the *many* attribute on this Schema.
- **kwargs** – Additional keyword arguments passed to *flask jsonify*.

Changed in version 0.6.0: Takes the same arguments as *marshmallow.Schema.dump*. Additional keyword arguments are passed to *flask jsonify*.

Changed in version 0.6.3: The *many* argument for this method defaults to the value of the *many* attribute on the Schema. Previously, the *many* argument of this method defaulted to False, regardless of the value of *Schema.many*.

load(*data*: Union[Mapping[str, Any], Iterable[Mapping[str, Any]]], *, *many*: Optional[bool] = None, *partial*: Optional[Union[bool, Sequence[str], Set[str]]] = None, *unknown*: Optional[str] = None)

Deserialize a data structure to an object defined by this Schema's fields.

Parameters

- **data** – The data to deserialize.
- **many** – Whether to deserialize *data* as a collection. If *None*, the value for *self.many* is used.
- **partial** – Whether to ignore missing fields and not require any fields declared. Propagates down to *Nested* fields as well. If its value is an iterable, only missing fields listed in that iterable will be ignored. Use dot delimiters to specify nested fields.
- **unknown** – Whether to exclude, include, or raise an error for unknown fields in the data. Use *EXCLUDE*, *INCLUDE* or *RAISE*. If *None*, the value for *self.unknown* is used.

Returns Deserialized data

New in version 1.0.0.

Changed in version 3.0.0b7: This method returns the deserialized data rather than a (data, errors) tuple. A *ValidationError* is raised if invalid data are passed.

loads(*json_data*: str, *, *many*: Optional[bool] = None, *partial*: Optional[Union[bool, Sequence[str], Set[str]]] = None, *unknown*: Optional[str] = None, ***kwargs*)

Same as *load()*, except it takes a JSON string as input.

Parameters

- **json_data** – A JSON string of the data to deserialize.
- **many** – Whether to deserialize *obj* as a collection. If *None*, the value for *self.many* is used.
- **partial** – Whether to ignore missing fields and not require any fields declared. Propagates down to *Nested* fields as well. If its value is an iterable, only missing fields listed in that iterable will be ignored. Use dot delimiters to specify nested fields.
- **unknown** – Whether to exclude, include, or raise an error for unknown fields in the data. Use *EXCLUDE*, *INCLUDE* or *RAISE*. If *None*, the value for *self.unknown* is used.

Returns Deserialized data

New in version 1.0.0.

Changed in version 3.0.0b7: This method returns the deserialized data rather than a (data, errors) tuple. A `ValidationError` is raised if invalid data are passed.

on_bind_field(*field_name: str, field_obj: marshmallow.fields.Field*) → None

Hook to modify a field when it is bound to the *Schema*.

No-op by default.

opts = <marshmallow.schema.SchemaOpts object>

property set_class: type

validate(*data: Union[Mapping[str, Any], Iterable[Mapping[str, Any]]], *, many: Optional[bool] = None, partial: Optional[Union[bool, Sequence[str], Set[str]]] = None*) → Dict[str, List[str]]

Validate *data* against the schema, returning a dictionary of validation errors.

Parameters

- **data** – The data to validate.
- **many** – Whether to validate *data* as a collection. If *None*, the value for *self.many* is used.
- **partial** – Whether to ignore missing fields and not require any fields declared. Propagates down to *Nested* fields as well. If its value is an iterable, only missing fields listed in that iterable will be ignored. Use dot delimiters to specify nested fields.

Returns A dictionary of validation errors.

New in version 1.1.0.

```
class app.serializers.core.TimestampField(*, load_default: Any = <marshmallow.missing>, missing: Any = <marshmallow.missing>, dump_default: Any = <marshmallow.missing>, default: Any = <marshmallow.missing>, data_key: Optional[str] = None, attribute: Optional[str] = None, validate: Optional[Union[Callable[[Any], Any], Iterable[Callable[[Any], Any]]]] = None, required: bool = False, allow_none: Optional[bool] = None, load_only: bool = False, dump_only: bool = False, error_messages: Optional[Dict[str, str]] = None, metadata: Optional[Mapping[str, Any]] = None, **additional_metadata)
```

Field that serializes to timestamp integer and deserializes to a `datetime.datetime` class.

_CHECK_ATTRIBUTE = True

_bind_to_schema(*field_name, schema*)

Update field with values from its parent schema. Called by `Schema._bind_field`.

Parameters

- **field_name** (*str*) – Field name set in schema.
- **schema** (*Schema / Field*) – Parent object.

_creation_index = 129

_deserialize(*value, attr, data, **kwargs*)

Deserialize value. Concrete **Field** classes should implement this method.

Parameters

- **value** – The value to be deserialized.
- **attr** – The attribute/key in *data* to be deserialized.
- **data** – The raw input data passed to the *Schema.load*.
- **kwargs** – Field-specific keyword arguments.

Raises **ValidationError** – In case of formatting or validation failure.

Returns The deserialized value.

Changed in version 2.0.0: Added **attr** and **data** parameters.

Changed in version 3.0.0: Added ****kwargs** to signature.

_serialize(*value*, *attr*, *obj*, ****kwargs**)

Serializes *value* to a basic Python datatype. Noop by default. Concrete Field classes should implement this method.

Example:

```
class TitleCase(Field):
    def _serialize(self, value, attr, obj, **kwargs):
        if not value:
            return ''
        return str(value).title()
```

Parameters

- **value** – The value to be serialized.
- **attr** (*str*) – The attribute or key on the object to be serialized.
- **obj** (*object*) – The object the value was pulled from.
- **kwargs** (*dict*) – Field-specific keyword arguments.

Returns The serialized value

_validate(*value*)

Perform validation on *value*. Raise a **ValidationError** if validation does not succeed.

property **_validate_all**

_validate_missing(*value*)

Validate missing values. Raise a **ValidationError** if *value* should be considered missing.

property **context**

The context dictionary for the parent Schema.

property **default**

default_error_messages = {'null': 'Field may not be null.', 'required': 'Missing data for required field.', 'validator_failed': 'Invalid value.'}

Default error messages for various kinds of errors. The keys in this dictionary are passed to *Field.make_error*. The values are error messages passed to *marshmallow.exceptions.ValidationError*.

deserialize(*value*: Any, *attr*: Optional[str] = None, *data*: Optional[Mapping[str, Any]] = None, ****kwargs**)
Deserialize value.

Parameters

- **value** – The value to deserialize.

- **attr** – The attribute/key in *data* to deserialize.
- **data** – The raw input data passed to *Schema.load*.
- **kwargs** – Field-specific keyword arguments.

Raises `ValidationError` – If an invalid value is passed or if a required value is missing.

fail(key: str, **kwargs)

Helper method that raises a *ValidationError* with an error message from `self.error_messages`.

Deprecated since version 3.0.0: Use `make_error <marshmallow.fields.Field.make_error>` instead.

get_value(obj, attr, accessor=None, default=<marshmallow.missing>)

Return the value for a given key from an object.

Parameters

- **obj** (object) – The object to get the value from.
- **attr** (str) – The attribute/key in *obj* to get the value from.
- **accessor** (callable) – A callable used to retrieve the value of *attr* from the object *obj*. Defaults to `marshmallow.utils.get_value`.

make_error(key: str, **kwargs) → `marshmallow.exceptions.ValidationError`

Helper method to make a *ValidationError* with an error message from `self.error_messages`.

property missing

name = None

parent = None

root = None

serialize(attr: str, obj: Any, accessor: Optional[Callable[[Any, str, Any], Any]] = None, **kwargs)

Pulls the value for the given key from the object, applies the field's formatting and returns the result.

Parameters

- **attr** – The attribute/key to get from the object.
- **obj** – The object to access the attribute/key from.
- **accessor** – Function used to access values from *obj*.
- **kwargs** – Field-specific keyword arguments.

```
class app.serializers.core._SearchOrderSerializer(*, only: Optional[Union[Sequence[str], Set[str]]]
    = None, exclude: Union[Sequence[str], Set[str]] =
    (), many: bool = False, context: Optional[Dict] =
    None, load_only: Union[Sequence[str], Set[str]] =
    (), dump_only: Union[Sequence[str], Set[str]] = (),
    partial: Union[bool, Sequence[str], Set[str]] =
    False, unknown: Optional[str] = None)
```

class Meta

Options object for a Schema.

Example usage:

```
class Meta:
    fields = ("id", "email", "date_created")
    exclude = ("password", "secret_attribute")
```

Available options:

- **fields:** Tuple or list of fields to include in the serialized result.
- **additional:** Tuple or list of fields to include *in addition to the* explicitly declared fields. **additional** and **fields** are mutually-exclusive options.
- **include:** Dictionary of additional fields to include in the schema. It is usually better to define fields as class variables, but you may need to use this option, e.g., if your fields are Python key-words. May be an *OrderedDict*.
- **exclude:** Tuple or list of fields to exclude in the serialized result. Nested fields can be represented with dot delimiters.
- **dateformat:** Default format for *Date* <fields.Date> fields.
- **datetimeformat:** Default format for *DateTime* <fields.DateTime> fields.
- **timeformat:** Default format for *Time* <fields.Time> fields.
- **render_module:** Module to use for *loads* <Schema.loads> and *dumps* <Schema.dumps>. Defaults to *json* from the standard library.
- **ordered:** If *True*, order serialization output according to the order in which fields were declared. Output of *Schema.dump* will be a *collections.OrderedDict*.
- **index_errors:** If *True*, errors dictionaries will include the **index** of invalid items in a collection.
- **load_only:** Tuple or list of fields to exclude from serialized results.
- **dump_only:** Tuple or list of fields to exclude from deserialization
- **unknown:** Whether to exclude, include, or raise an error for unknown fields in the data. Use *EXCLUDE*, *INCLUDE* or *RAISE*.
- **register:** Whether to register the *Schema* with *marshmallow*'s internal class registry. Must be *True* if you intend to refer to this *Schema* by class name in *Nested* fields. Only set this to *False* when memory usage is critical. Defaults to *True*.

OPTIONS_CLASS

alias of `marshmallow.schema.SchemaOpts`

```
TYPE_MAPPING = {<class 'str':>: <class 'marshmallow.fields.String'>, <class
'bytes':>: <class 'marshmallow.fields.String'>, <class 'datetime.datetime':>: <class
'marshmallow.fields.DateTime'>, <class 'float':>: <class
'marshmallow.fields.Float'>, <class 'bool':>: <class 'marshmallow.fields.Boolean'>,
<class 'tuple':>: <class 'marshmallow.fields.Raw'>, <class 'list':>: <class
'marshmallow.fields.Raw'>, <class 'set':>: <class 'marshmallow.fields.Raw'>, <class
'int':>: <class 'marshmallow.fields.Integer'>, <class 'uuid.UUID':>: <class
'marshmallow.fields.UUID'>, <class 'datetime.time':>: <class
'marshmallow.fields.Time'>, <class 'datetime.date':>: <class
'marshmallow.fields.Date'>, <class 'datetime.timedelta':>: <class
'marshmallow.fields.TimeDelta'>, <class 'decimal.Decimal':>: <class
'marshmallow.fields.Decimal'>}
```

_bind_field(*field_name*: str, *field_obj*: marshmallow.fields.Field) → None

Bind field to the schema, setting any necessary attributes on the field (e.g. parent and name).

Also set field *load_only* and *dump_only* values if *field_name* was specified in class *Meta*.

static _call_and_store(*getter_func*, *data*, *, *field_name*, *error_store*, *index*=None)

Call *getter_func* with *data* as its argument, and store any *ValidationErrors*.

Parameters

- **getter_func** (*callable*) – Function for getting the serialized/deserialized value from data.
- **data** – The data passed to `getter_func`.
- **field_name** (*str*) – Field name.
- **index** (*int*) – Index of the item being validated, if validating a collection, otherwise *None*.

```
_declared_fields = {'field_name':
<fields.String(dump_default=<marshmallow.missing>, attribute=None, validate=None,
required=False, load_only=False, dump_only=False,
load_default=<marshmallow.missing>, allow_none=False, error_messages={'required':
'Missing data for required field.', 'null': 'Field may not be null.',
'validator_failed': 'Invalid value.', 'invalid': 'Not a valid string.',
'invalid_utf8': 'Not a valid utf-8 string.'})>, 'sorting':
<fields.String(dump_default=<marshmallow.missing>, attribute=None,
validate=<OneOf(choices=['asc', 'desc'], labels=[], error='Must be one of:
{choices}.')>, required=False, load_only=False, dump_only=False,
load_default=<marshmallow.missing>, allow_none=False, error_messages={'required':
'Missing data for required field.', 'null': 'Field may not be null.',
'validator_failed': 'Invalid value.', 'invalid': 'Not a valid string.',
'invalid_utf8': 'Not a valid utf-8 string.'})>>}
```

```
_default_error_messages = {'type': 'Invalid input type.', 'unknown': 'Unknown
field.'}
```

```
_deserialize(data: Union[Mapping[str, Any], Iterable[Mapping[str, Any]]], *, error_store:
marshmallow.error_store.ErrorStore, many: bool = False, partial=False, unknown='raise',
index=None) → Union[marshmallow.schema._T, List[marshmallow.schema._T]]
```

Deserialize data.

Parameters

- **data** (*dict*) – The data to deserialize.
- **error_store** (*ErrorStore*) – Structure to store errors.
- **many** (*bool*) – *True* if data should be deserialized as a collection.
- **partial** (*bool | tuple*) – Whether to ignore missing fields and not require any fields declared. Propagates down to Nested fields as well. If its value is an iterable, only missing fields listed in that iterable will be ignored. Use dot delimiters to specify nested fields.
- **unknown** – Whether to exclude, include, or raise an error for unknown fields in the data. Use *EXCLUDE*, *INCLUDE* or *RAISE*.
- **index** (*int*) – Index of the item being serialized (for storing errors) if serializing a collection, otherwise *None*.

Returns A dictionary of the deserialized data.

```
_do_load(data: Union[Mapping[str, Any], Iterable[Mapping[str, Any]]], *, many: Optional[bool] = None,
partial: Optional[Union[bool, Sequence[str], Set[str]]] = None, unknown: Optional[str] = None,
postprocess: bool = True)
```

Deserialize *data*, returning the deserialized result. This method is private API.

Parameters

- **data** – The data to deserialize.
- **many** – Whether to deserialize *data* as a collection. If *None*, the value for *self.many* is used.

- **partial** – Whether to validate required fields. If its value is an iterable, only fields listed in that iterable will be ignored will be allowed missing. If *True*, all fields will be allowed missing. If *None*, the value for *self.partial* is used.
- **unknown** – Whether to exclude, include, or raise an error for unknown fields in the data. Use *EXCLUDE*, *INCLUDE* or *RAISE*. If *None*, the value for *self.unknown* is used.
- **postprocess** – Whether to run *post_load* methods..

Returns Deserialized data

_has_processors(*tag*) → bool

_hooks = {}

_init_fields() → None

Update *self.fields*, *self.load_fields*, and *self.dump_fields* based on schema options. This method is private API.

_invoke_dump_processors(*tag: str*, *data*, *, *many: bool*, *original_data=None*)

_invoke_field_validators(*, *error_store: marshmallow.error_store.ErrorStore*, *data*, *many: bool*)

_invoke_load_processors(*tag: str*, *data*, *, *many: bool*, *original_data*, *partial: Union[bool, Sequence[str], Set[str]]*)

_invoke_processors(*tag: str*, *, *pass_many: bool*, *data*, *many: bool*, *original_data=None*, ***kwargs*)

_invoke_schema_validators(*, *error_store: marshmallow.error_store.ErrorStore*, *pass_many: bool*, *data*, *original_data*, *many: bool*, *partial: Union[bool, Sequence[str], Set[str]]*, *field_errors: bool = False*)

_normalize_nested_options() → None

Apply then flatten nested schema options. This method is private API.

_run_validator(*validator_func*, *output*, *, *original_data*, *error_store*, *many*, *partial*, *pass_original*, *index=None*)

_serialize(*obj: Union[marshmallow.schema._T, Iterable[marshmallow.schema._T]]*, *, *many: bool = False*)

Serialize *obj*.

Parameters

- **obj** – The object(s) to serialize.
- **many** (*bool*) – *True* if data should be serialized as a collection.

Returns A dictionary of the serialized data

Changed in version 1.0.0: Renamed from *marshal*.

property dict_class: **type**

dump(*obj: Any*, *, *many: Optional[bool] = None*)

Serialize an object to native Python data types according to this Schema's fields.

Parameters

- **obj** – The object to serialize.
- **many** – Whether to serialize *obj* as a collection. If *None*, the value for *self.many* is used.

Returns Serialized data

New in version 1.0.0.

Changed in version 3.0.0b7: This method returns the serialized data rather than a (data, errors) tuple. A `ValidationError` is raised if `obj` is invalid.

Changed in version 3.0.0rc9: Validation no longer occurs upon serialization.

dumps(*obj: Any, *args, many: Optional[bool] = None, **kwargs*)

Same as `dump()`, except return a JSON-encoded string.

Parameters

- **obj** – The object to serialize.
- **many** – Whether to serialize *obj* as a collection. If *None*, the value for *self.many* is used.

Returns A json string

New in version 1.0.0.

Changed in version 3.0.0b7: This method returns the serialized data rather than a (data, errors) tuple. A `ValidationError` is raised if `obj` is invalid.

error_messages = {}

Overrides for default schema-level error messages

fields

Dictionary mapping field_names -> Field objects

classmethod from_dict(*fields: Dict[str, Union[marshmallow.fields.Field, type]], *, name: str = 'GeneratedSchema'*) → type

Generate a *Schema* class given a dictionary of fields.

```
from marshmallow import Schema, fields

PersonSchema = Schema.from_dict({"name": fields.Str()})
print(PersonSchema().load({"name": "David"})) # => {'name': 'David'}
```

Generated schemas are not added to the class registry and therefore cannot be referred to by name in *Nested* fields.

Parameters

- **fields** (*dict*) – Dictionary mapping field names to field instances.
- **name** (*str*) – Optional name for the class, which will appear in the `repr` for the class.

New in version 3.0.0.

get_attribute(*obj: Any, attr: str, default: Any*)

Defines how to pull values from an object to serialize.

New in version 2.0.0.

Changed in version 3.0.0a1: Changed position of `obj` and `attr`.

handle_error(*error: marshmallow.exceptions.ValidationError, data: Any, *, many: bool, **kwargs*)

Custom error handler function for the schema.

Parameters

- **error** – The *ValidationError* raised during (de)serialization.
- **data** – The original input data.
- **many** – Value of `many` on `dump` or `load`.

- **partial** – Value of **partial** on load.

New in version 2.0.0.

Changed in version 3.0.0rc9: Receives *many* and *partial* (on deserialization) as keyword arguments.

jsonify(*obj*, *many*=<object object>, **args*, ***kwargs*)
Return a JSON response containing the serialized data.

Parameters

- **obj** – Object to serialize.
- **many** (*bool*) – Whether *obj* should be serialized as an instance or as a collection. If unset, defaults to the value of the *many* attribute on this Schema.
- **kwargs** – Additional keyword arguments passed to *flask jsonify*.

Changed in version 0.6.0: Takes the same arguments as *marshmallow.Schema.dump*. Additional keyword arguments are passed to *flask jsonify*.

Changed in version 0.6.3: The *many* argument for this method defaults to the value of the *many* attribute on the Schema. Previously, the *many* argument of this method defaulted to False, regardless of the value of *Schema.many*.

load(*data*: Union[Mapping[str, Any], Iterable[Mapping[str, Any]]], *, *many*: Optional[bool] = None, *partial*: Optional[Union[bool, Sequence[str], Set[str]]] = None, *unknown*: Optional[str] = None)
Deserialize a data structure to an object defined by this Schema's fields.

Parameters

- **data** – The data to deserialize.
- **many** – Whether to deserialize *data* as a collection. If *None*, the value for *self.many* is used.
- **partial** – Whether to ignore missing fields and not require any fields declared. Propagates down to *Nested* fields as well. If its value is an iterable, only missing fields listed in that iterable will be ignored. Use dot delimiters to specify nested fields.
- **unknown** – Whether to exclude, include, or raise an error for unknown fields in the data. Use *EXCLUDE*, *INCLUDE* or *RAISE*. If *None*, the value for *self.unknown* is used.

Returns Deserialized data

New in version 1.0.0.

Changed in version 3.0.0b7: This method returns the deserialized data rather than a (data, errors) tuple. A *ValidationError* is raised if invalid data are passed.

loads(*json_data*: str, *, *many*: Optional[bool] = None, *partial*: Optional[Union[bool, Sequence[str], Set[str]]] = None, *unknown*: Optional[str] = None, ***kwargs*)
Same as *load()*, except it takes a JSON string as input.

Parameters

- **json_data** – A JSON string of the data to deserialize.
- **many** – Whether to deserialize *obj* as a collection. If *None*, the value for *self.many* is used.
- **partial** – Whether to ignore missing fields and not require any fields declared. Propagates down to *Nested* fields as well. If its value is an iterable, only missing fields listed in that iterable will be ignored. Use dot delimiters to specify nested fields.
- **unknown** – Whether to exclude, include, or raise an error for unknown fields in the data. Use *EXCLUDE*, *INCLUDE* or *RAISE*. If *None*, the value for *self.unknown* is used.

Returns Deserialized data

New in version 1.0.0.

Changed in version 3.0.0b7: This method returns the deserialized data rather than a (data, errors) tuple. A `ValidationError` is raised if invalid data are passed.

on_bind_field(*field_name: str, field_obj: marshmallow.fields.Field*) → None

Hook to modify a field when it is bound to the *Schema*.

No-op by default.

opts = <marshmallow.schema.SchemaOpts object>

property set_class: type

validate(*data: Union[Mapping[str, Any], Iterable[Mapping[str, Any]]], *, many: Optional[bool] = None, partial: Optional[Union[bool, Sequence[str], Set[str]]] = None*) → Dict[str, List[str]]

Validate *data* against the schema, returning a dictionary of validation errors.

Parameters

- **data** – The data to validate.
- **many** – Whether to validate *data* as a collection. If *None*, the value for *self.many* is used.
- **partial** – Whether to ignore missing fields and not require any fields declared. Propagates down to *Nested* fields as well. If its value is an iterable, only missing fields listed in that iterable will be ignored. Use dot delimiters to specify nested fields.

Returns A dictionary of validation errors.

New in version 1.1.0.

```
class app.serializers.core._SearchValueSerializer(*, only: Optional[Union[Sequence[str], Set[str]]] = None, exclude: Union[Sequence[str], Set[str]] = (), many: bool = False, context: Optional[Dict] = None, load_only: Union[Sequence[str], Set[str]] = (), dump_only: Union[Sequence[str], Set[str]] = (), partial: Union[bool, Sequence[str], Set[str]] = False, unknown: Optional[str] = None)
```

class Meta

Options object for a *Schema*.

Example usage:

```
class Meta:
    fields = ("id", "email", "date_created")
    exclude = ("password", "secret_attribute")
```

Available options:

- **fields:** Tuple or list of fields to include in the serialized result.
- **additional:** Tuple or list of fields to include in addition to the explicitly declared fields. **additional** and **fields** are mutually-exclusive options.
- **include:** Dictionary of additional fields to include in the schema. **It is** usually better to define fields as class variables, but you may need to use this option, e.g., if your fields are Python key-words. May be an *OrderedDict*.

- **exclude:** Tuple or list of fields to exclude in the serialized result. Nested fields can be represented with dot delimiters.
- **dateformat:** Default format for *Date* <fields.Date> fields.
- **datetimeformat:** Default format for *DateTime* <fields.DateTime> fields.
- **timeformat:** Default format for *Time* <fields.Time> fields.
- **render_module:** Module to use for *loads* <Schema.loads> and *dumps* <Schema.dumps>. Defaults to *json* from the standard library.
- **ordered:** If *True*, order serialization output according to the order in which fields were declared. Output of *Schema.dump* will be a *collections.OrderedDict*.
- **index_errors:** If *True*, errors dictionaries will include the **index** of invalid items in a collection.
- **load_only:** Tuple or list of fields to exclude from serialized results.
- **dump_only:** Tuple or list of fields to exclude from deserialization
- **unknown:** Whether to exclude, include, or raise an error for unknown fields in the data. Use *EXCLUDE*, *INCLUDE* or *RAISE*.
- **register:** Whether to register the *Schema* with *marshmallow*'s internal class registry. Must be *True* if you intend to refer to this *Schema* by class name in *Nested* fields. Only set this to *False* when memory usage is critical. Defaults to *True*.

OPTIONS_CLASS

alias of `marshmallow.schema.SchemaOpts`

```
TYPE_MAPPING = {<class 'str':>: <class 'marshmallow.fields.String'>, <class
'bytes':>: <class 'marshmallow.fields.String'>, <class 'datetime.datetime':>: <class
'marshmallow.fields.DateTime'>, <class 'float':>: <class
'marshmallow.fields.Float'>, <class 'bool':>: <class 'marshmallow.fields.Boolean'>,
<class 'tuple':>: <class 'marshmallow.fields.Raw'>, <class 'list':>: <class
'marshmallow.fields.Raw'>, <class 'set':>: <class 'marshmallow.fields.Raw'>, <class
'int':>: <class 'marshmallow.fields.Integer'>, <class 'uuid.UUID':>: <class
'marshmallow.fields.UUID'>, <class 'datetime.time':>: <class
'marshmallow.fields.Time'>, <class 'datetime.date':>: <class
'marshmallow.fields.Date'>, <class 'datetime.timedelta':>: <class
'marshmallow.fields.TimeDelta'>, <class 'decimal.Decimal':>: <class
'marshmallow.fields.Decimal'>}
```

_bind_field(*field_name*: str, *field_obj*: marshmallow.fields.Field) → None

Bind field to the schema, setting any necessary attributes on the field (e.g. parent and name).

Also set field *load_only* and *dump_only* values if *field_name* was specified in class *Meta*.

static _call_and_store(*getter_func*, *data*, *, *field_name*, *error_store*, *index=None*)

Call *getter_func* with *data* as its argument, and store any *ValidationErrors*.

Parameters

- **getter_func** (callable) – Function for getting the serialized/deserialized value from data.
- **data** – The data passed to *getter_func*.
- **field_name** (str) – Field name.
- **index** (int) – Index of the item being validated, if validating a collection, otherwise *None*.

```

_declared_fields = {'field_name':
<fields.String(dump_default=<marshmallow.missing>, attribute=None, validate=None,
required=False, load_only=False, dump_only=False,
load_default=<marshmallow.missing>, allow_none=False, error_messages={'required':
'Missing data for required field.', 'null': 'Field may not be null.',
'validator_failed': 'Invalid value.', 'invalid': 'Not a valid string.',
'invalid_utf8': 'Not a valid utf-8 string.'})>, 'field_operator':
<fields.String(dump_default=<marshmallow.missing>, attribute=None,
validate=<OneOf(choices={'in', 'gt', 'eq', 'gte', 'lte', 'nin', 'contains',
'startswith', 'ne', 'between', 'endswith', 'lt', 'ncontains'}, labels=[]),
error='Must be one of: {choices}.')>, required=False, load_only=False,
dump_only=False, load_default=<marshmallow.missing>, allow_none=False,
error_messages={'required': 'Missing data for required field.', 'null': 'Field may
not be null.', 'validator_failed': 'Invalid value.', 'invalid': 'Not a valid
string.', 'invalid_utf8': 'Not a valid utf-8 string.'})>, 'field_value':
<fields.Raw(dump_default=<marshmallow.missing>, attribute=None, validate=None,
required=False, load_only=False, dump_only=False,
load_default=<marshmallow.missing>, allow_none=False, error_messages={'required':
'Missing data for required field.', 'null': 'Field may not be null.',
'validator_failed': 'Invalid value.'})>>

_default_error_messages = {'type': 'Invalid input type.', 'unknown': 'Unknown
field.'}

_deserialize(data: Union[Mapping[str, Any], Iterable[Mapping[str, Any]]], *, error_store:
marshmallow.error_store.ErrorStore, many: bool = False, partial=False, unknown='raise',
index=None) → Union[marshmallow.schema._T, List[marshmallow.schema._T]]

```

Deserialize data.

Parameters

- **data** (*dict*) – The data to deserialize.
- **error_store** (*ErrorStore*) – Structure to store errors.
- **many** (*bool*) – *True* if data should be deserialized as a collection.
- **partial** (*bool | tuple*) – Whether to ignore missing fields and not require any fields declared. Propagates down to Nested fields as well. If its value is an iterable, only missing fields listed in that iterable will be ignored. Use dot delimiters to specify nested fields.
- **unknown** – Whether to exclude, include, or raise an error for unknown fields in the data. Use *EXCLUDE*, *INCLUDE* or *RAISE*.
- **index** (*int*) – Index of the item being serialized (for storing errors) if serializing a collection, otherwise *None*.

Returns A dictionary of the deserialized data.

```

_do_load(data: Union[Mapping[str, Any], Iterable[Mapping[str, Any]]], *, many: Optional[bool] = None,
partial: Optional[Union[bool, Sequence[str], Set[str]]] = None, unknown: Optional[str] = None,
postprocess: bool = True)

```

Deserialize *data*, returning the deserialized result. This method is private API.

Parameters

- **data** – The data to deserialize.
- **many** – Whether to deserialize *data* as a collection. If *None*, the value for *self.many* is used.

- **partial** – Whether to validate required fields. If its value is an iterable, only fields listed in that iterable will be ignored will be allowed missing. If *True*, all fields will be allowed missing. If *None*, the value for *self.partial* is used.
- **unknown** – Whether to exclude, include, or raise an error for unknown fields in the data. Use *EXCLUDE*, *INCLUDE* or *RAISE*. If *None*, the value for *self.unknown* is used.
- **postprocess** – Whether to run *post_load* methods..

Returns Deserialized data

_has_processors(*tag*) → bool

_hooks = {}

_init_fields() → None

Update *self.fields*, *self.load_fields*, and *self.dump_fields* based on schema options. This method is private API.

_invoke_dump_processors(*tag: str*, *data*, *, *many: bool*, *original_data=None*)

_invoke_field_validators(*, *error_store: marshmallow.error_store.ErrorStore*, *data*, *many: bool*)

_invoke_load_processors(*tag: str*, *data*, *, *many: bool*, *original_data*, *partial: Union[bool, Sequence[str], Set[str]]*)

_invoke_processors(*tag: str*, *, *pass_many: bool*, *data*, *many: bool*, *original_data=None*, **kwargs)

_invoke_schema_validators(*, *error_store: marshmallow.error_store.ErrorStore*, *pass_many: bool*, *data*, *original_data*, *many: bool*, *partial: Union[bool, Sequence[str], Set[str]]*, *field_errors: bool = False*)

_normalize_nested_options() → None

Apply then flatten nested schema options. This method is private API.

_run_validator(*validator_func*, *output*, *, *original_data*, *error_store*, *many*, *partial*, *pass_original*, *index=None*)

_serialize(*obj: Union[marshmallow.schema._T, Iterable[marshmallow.schema._T]]*, *, *many: bool = False*)

Serialize *obj*.

Parameters

- **obj** – The object(s) to serialize.
- **many** (*bool*) – *True* if data should be serialized as a collection.

Returns A dictionary of the serialized data

Changed in version 1.0.0: Renamed from *marshal*.

property dict_class: type

dump(*obj: Any*, *, *many: Optional[bool] = None*)

Serialize an object to native Python data types according to this Schema's fields.

Parameters

- **obj** – The object to serialize.
- **many** – Whether to serialize *obj* as a collection. If *None*, the value for *self.many* is used.

Returns Serialized data

New in version 1.0.0.

Changed in version 3.0.0b7: This method returns the serialized data rather than a (data, errors) tuple. A `ValidationError` is raised if `obj` is invalid.

Changed in version 3.0.0rc9: Validation no longer occurs upon serialization.

dumps(*obj: Any, *args, many: Optional[bool] = None, **kwargs*)

Same as `dump()`, except return a JSON-encoded string.

Parameters

- **obj** – The object to serialize.
- **many** – Whether to serialize *obj* as a collection. If *None*, the value for *self.many* is used.

Returns A json string

New in version 1.0.0.

Changed in version 3.0.0b7: This method returns the serialized data rather than a (data, errors) tuple. A `ValidationError` is raised if `obj` is invalid.

error_messages = {}

Overrides for default schema-level error messages

fields

Dictionary mapping field_names -> Field objects

classmethod from_dict(*fields: Dict[str, Union[marshmallow.fields.Field, type]], *, name: str = 'GeneratedSchema'*) → type

Generate a *Schema* class given a dictionary of fields.

```
from marshmallow import Schema, fields

PersonSchema = Schema.from_dict({"name": fields.Str()})
print(PersonSchema().load({"name": "David"})) # => {'name': 'David'}
```

Generated schemas are not added to the class registry and therefore cannot be referred to by name in *Nested* fields.

Parameters

- **fields** (*dict*) – Dictionary mapping field names to field instances.
- **name** (*str*) – Optional name for the class, which will appear in the `repr` for the class.

New in version 3.0.0.

get_attribute(*obj: Any, attr: str, default: Any*)

Defines how to pull values from an object to serialize.

New in version 2.0.0.

Changed in version 3.0.0a1: Changed position of `obj` and `attr`.

handle_error(*error: marshmallow.exceptions.ValidationError, data: Any, *, many: bool, **kwargs*)

Custom error handler function for the schema.

Parameters

- **error** – The *ValidationError* raised during (de)serialization.
- **data** – The original input data.
- **many** – Value of `many` on `dump` or `load`.

- **partial** – Value of **partial** on load.

New in version 2.0.0.

Changed in version 3.0.0rc9: Receives *many* and *partial* (on deserialization) as keyword arguments.

jsonify(*obj*, *many*=<object object>, **args*, ***kwargs*)
Return a JSON response containing the serialized data.

Parameters

- **obj** – Object to serialize.
- **many** (*bool*) – Whether *obj* should be serialized as an instance or as a collection. If unset, defaults to the value of the *many* attribute on this Schema.
- **kwargs** – Additional keyword arguments passed to *flask jsonify*.

Changed in version 0.6.0: Takes the same arguments as *marshmallow.Schema.dump*. Additional keyword arguments are passed to *flask jsonify*.

Changed in version 0.6.3: The *many* argument for this method defaults to the value of the *many* attribute on the Schema. Previously, the *many* argument of this method defaulted to False, regardless of the value of *Schema.many*.

load(*data*: Union[Mapping[str, Any], Iterable[Mapping[str, Any]]], *, *many*: Optional[bool] = None, *partial*: Optional[Union[bool, Sequence[str], Set[str]]] = None, *unknown*: Optional[str] = None)
Deserialize a data structure to an object defined by this Schema's fields.

Parameters

- **data** – The data to deserialize.
- **many** – Whether to deserialize *data* as a collection. If *None*, the value for *self.many* is used.
- **partial** – Whether to ignore missing fields and not require any fields declared. Propagates down to *Nested* fields as well. If its value is an iterable, only missing fields listed in that iterable will be ignored. Use dot delimiters to specify nested fields.
- **unknown** – Whether to exclude, include, or raise an error for unknown fields in the data. Use *EXCLUDE*, *INCLUDE* or *RAISE*. If *None*, the value for *self.unknown* is used.

Returns Deserialized data

New in version 1.0.0.

Changed in version 3.0.0b7: This method returns the deserialized data rather than a (data, errors) tuple. A *ValidationError* is raised if invalid data are passed.

loads(*json_data*: str, *, *many*: Optional[bool] = None, *partial*: Optional[Union[bool, Sequence[str], Set[str]]] = None, *unknown*: Optional[str] = None, ***kwargs*)
Same as *load()*, except it takes a JSON string as input.

Parameters

- **json_data** – A JSON string of the data to deserialize.
- **many** – Whether to deserialize *obj* as a collection. If *None*, the value for *self.many* is used.
- **partial** – Whether to ignore missing fields and not require any fields declared. Propagates down to *Nested* fields as well. If its value is an iterable, only missing fields listed in that iterable will be ignored. Use dot delimiters to specify nested fields.
- **unknown** – Whether to exclude, include, or raise an error for unknown fields in the data. Use *EXCLUDE*, *INCLUDE* or *RAISE*. If *None*, the value for *self.unknown* is used.

Returns Deserialized data

New in version 1.0.0.

Changed in version 3.0.0b7: This method returns the deserialized data rather than a (data, errors) tuple. A `ValidationError` is raised if invalid data are passed.

on_bind_field(*field_name: str, field_obj: marshmallow.fields.Field*) → None

Hook to modify a field when it is bound to the *Schema*.

No-op by default.

opts = <marshmallow.schema.SchemaOpts object>

property set_class: type

validate(*data: Union[Mapping[str, Any], Iterable[Mapping[str, Any]]], *, many: Optional[bool] = None, partial: Optional[Union[bool, Sequence[str], Set[str]]] = None*) → Dict[str, List[str]]

Validate *data* against the schema, returning a dictionary of validation errors.

Parameters

- **data** – The data to validate.
- **many** – Whether to validate *data* as a collection. If *None*, the value for *self.many* is used.
- **partial** – Whether to ignore missing fields and not require any fields declared. Propagates down to *Nested* fields as well. If its value is an iterable, only missing fields listed in that iterable will be ignored. Use dot delimiters to specify nested fields.

Returns A dictionary of validation errors.

New in version 1.1.0.

app.serializers.document

Description

Classes

`DocumentAttachmentSerializer(*[, only, ...])`

`DocumentSerializer(*[, only, exclude, many, ...])`

app.serializers.document.DocumentAttachmentSerializer

```
class app.serializers.document.DocumentAttachmentSerializer(*, only:
    Optional[Union[Sequence[str],
    Set[str]]] = None, exclude:
    Union[Sequence[str], Set[str]] = (),
    many: bool = False, context:
    Optional[Dict] = None, load_only:
    Union[Sequence[str], Set[str]] = (),
    dump_only: Union[Sequence[str],
    Set[str]] = (), partial: Union[bool,
    Sequence[str], Set[str]] = False,
    unknown: Optional[str] = None)
```

Bases: flask_marshmallow.schema.Schema

Attributes

DocumentAttachmentSerializer.

TYPE_MAPPING

DocumentAttachmentSerializer.

dict_class

DocumentAttachmentSerializer.

error_messages

Overrides for default schema-level error mes-
sages

DocumentAttachmentSerializer.opts

DocumentAttachmentSerializer.

set_class

app.serializers.document.DocumentAttachmentSerializer.TYPE_MAPPING

```
DocumentAttachmentSerializer.TYPE_MAPPING = {<class 'str':>: <class
'marshmallow.fields.String'>, <class 'bytes':>: <class
'marshmallow.fields.String'>, <class 'datetime.datetime':>: <class
'marshmallow.fields.DateTime'>, <class 'float':>: <class
'marshmallow.fields.Float'>, <class 'bool':>: <class
'marshmallow.fields.Boolean'>, <class 'tuple':>: <class
'marshmallow.fields.Raw'>, <class 'list':>: <class
'marshmallow.fields.Raw'>, <class 'set':>: <class 'marshmallow.fields.Raw'>,
<class 'int':>: <class 'marshmallow.fields.Integer'>, <class 'uuid.UUID':>:
<class 'marshmallow.fields.UUID'>, <class 'datetime.time':>: <class
'marshmallow.fields.Time'>, <class 'datetime.date':>: <class
'marshmallow.fields.Date'>, <class 'datetime.timedelta':>: <class
'marshmallow.fields.TimeDelta'>, <class 'decimal.Decimal':>: <class
'marshmallow.fields.Decimal'>}
```

app.serializers.document.DocumentAttachmentSerializer.dict_class

property DocumentAttachmentSerializer.dict_class: type

app.serializers.document.DocumentAttachmentSerializer.error_messages

DocumentAttachmentSerializer.error_messages = {}
 Overrides for default schema-level error messages

app.serializers.document.DocumentAttachmentSerializer.opts

DocumentAttachmentSerializer.opts = <marshmallow.schema.SchemaOpts object>

app.serializers.document.DocumentAttachmentSerializer.set_class

property DocumentAttachmentSerializer.set_class: type

Methods

<i>DocumentAttachmentSerializer.__init__(*[, ...])</i>	
<i>DocumentAttachmentSerializer.dump(obj, *[, many])</i>	Serialize an object to native Python data types according to this Schema's fields.
<i>DocumentAttachmentSerializer.dumps(obj, *args)</i>	Same as <i>dump()</i> , except return a JSON-encoded string.
<i>DocumentAttachmentSerializer.from_dict(fields, *)</i>	Generate a <i>Schema</i> class given a dictionary of fields.
<i>DocumentAttachmentSerializer.get_attribute(...)</i>	Defines how to pull values from an object to serialize.
<i>DocumentAttachmentSerializer.handle_error(...)</i>	Custom error handler function for the schema.
<i>DocumentAttachmentSerializer jsonify(obj[, many])</i>	Return a JSON response containing the serialized data.
<i>DocumentAttachmentSerializer.load(data, *[, ...])</i>	Deserialize a data structure to an object defined by this Schema's fields.
<i>DocumentAttachmentSerializer.loads(json_data, *)</i>	Same as <i>load()</i> , except it takes a JSON string as input.
<i>DocumentAttachmentSerializer.on_bind_field(...)</i>	Hook to modify a field when it is bound to the <i>Schema</i> .
<i>DocumentAttachmentSerializer.process_input(...)</i>	
<i>DocumentAttachmentSerializer.validate(data, *)</i>	Validate <i>data</i> against the schema, returning a dictionary of validation errors.

app.serializers.document.DocumentAttachmentSerializer.__init__

```
DocumentAttachmentSerializer.__init__(*, only: Optional[Union[Sequence[str], Set[str]]]
                                     = None, exclude: Union[Sequence[str], Set[str]]
                                     = (), many: bool = False, context: Optional[Dict]
                                     = None, load_only: Union[Sequence[str],
                                     Set[str]] = (), dump_only: Union[Sequence[str],
                                     Set[str]] = (), partial: Union[bool, Sequence[str],
                                     Set[str]] = False, unknown: Optional[str] =
                                     None)
```

app.serializers.document.DocumentAttachmentSerializer.dump

```
DocumentAttachmentSerializer.dump(obj: Any, *, many: Optional[bool] = None)
```

Serialize an object to native Python data types according to this Schema's fields.

Parameters

- **obj** – The object to serialize.
- **many** – Whether to serialize *obj* as a collection. If *None*, the value for *self.many* is used.

Returns Serialized data

New in version 1.0.0.

Changed in version 3.0.0b7: This method returns the serialized data rather than a (data, errors) tuple. A `ValidationError` is raised if *obj* is invalid.

Changed in version 3.0.0rc9: Validation no longer occurs upon serialization.

app.serializers.document.DocumentAttachmentSerializer.dumps

```
DocumentAttachmentSerializer.dumps(obj: Any, *args, many: Optional[bool] = None,
                                   **kwargs)
```

Same as `dump()`, except return a JSON-encoded string.

Parameters

- **obj** – The object to serialize.
- **many** – Whether to serialize *obj* as a collection. If *None*, the value for *self.many* is used.

Returns A json string

New in version 1.0.0.

Changed in version 3.0.0b7: This method returns the serialized data rather than a (data, errors) tuple. A `ValidationError` is raised if *obj* is invalid.

app.serializers.document.DocumentAttachmentSerializer.from_dict

```
classmethod DocumentAttachmentSerializer.from_dict(fields: Dict[str,
                                                             Union[marshmallow.fields.Field,
                                                             type]], *, name: str =
                                                             'GeneratedSchema') → type
```

Generate a *Schema* class given a dictionary of fields.

```
from marshmallow import Schema, fields

PersonSchema = Schema.from_dict({"name": fields.Str()})
print(PersonSchema().load({"name": "David"})) # => {'name': 'David'}
```

Generated schemas are not added to the class registry and therefore cannot be referred to by name in *Nested* fields.

Parameters

- **fields** (*dict*) – Dictionary mapping field names to field instances.
- **name** (*str*) – Optional name for the class, which will appear in the repr for the class.

New in version 3.0.0.

app.serializers.document.DocumentAttachmentSerializer.get_attribute

`DocumentAttachmentSerializer.get_attribute(obj: Any, attr: str, default: Any)`

Defines how to pull values from an object to serialize.

New in version 2.0.0.

Changed in version 3.0.0a1: Changed position of *obj* and *attr*.

app.serializers.document.DocumentAttachmentSerializer.handle_error

`DocumentAttachmentSerializer.handle_error(error: marshmallow.exceptions.ValidationError, data: Any, *, many: bool, **kwargs)`

Custom error handler function for the schema.

Parameters

- **error** – The *ValidationError* raised during (de)serialization.
- **data** – The original input data.
- **many** – Value of *many* on dump or load.
- **partial** – Value of *partial* on load.

New in version 2.0.0.

Changed in version 3.0.0rc9: Receives *many* and *partial* (on deserialization) as keyword arguments.

app.serializers.document.DocumentAttachmentSerializer.jsonify

`DocumentAttachmentSerializer.jsonify(obj, many=<object object>, *args, **kwargs)`

Return a JSON response containing the serialized data.

Parameters

- **obj** – Object to serialize.
- **many** (*bool*) – Whether *obj* should be serialized as an instance or as a collection. If unset, defaults to the value of the *many* attribute on this Schema.
- **kwargs** – Additional keyword arguments passed to *flask jsonify*.

Changed in version 0.6.0: Takes the same arguments as *marshmallow.Schema.dump*. Additional keyword arguments are passed to *flask jsonify*.

Changed in version 0.6.3: The *many* argument for this method defaults to the value of the *many* attribute on the Schema. Previously, the *many* argument of this method defaulted to False, regardless of the value of *Schema.many*.

app.serializers.document.DocumentAttachmentSerializer.load

`DocumentAttachmentSerializer.load(data: Union[Mapping[str, Any], Iterable[Mapping[str, Any]]], *, many: Optional[bool] = None, partial: Optional[Union[bool, Sequence[str], Set[str]]] = None, unknown: Optional[str] = None)`

Deserialize a data structure to an object defined by this Schema's fields.

Parameters

- **data** – The data to deserialize.
- **many** – Whether to deserialize *data* as a collection. If *None*, the value for *self.many* is used.
- **partial** – Whether to ignore missing fields and not require any fields declared. Propagates down to *Nested* fields as well. If its value is an iterable, only missing fields listed in that iterable will be ignored. Use dot delimiters to specify nested fields.
- **unknown** – Whether to exclude, include, or raise an error for unknown fields in the data. Use *EXCLUDE*, *INCLUDE* or *RAISE*. If *None*, the value for *self.unknown* is used.

Returns Deserialized data

New in version 1.0.0.

Changed in version 3.0.0b7: This method returns the deserialized data rather than a (data, errors) tuple. A `ValidationError` is raised if invalid data are passed.

app.serializers.document.DocumentAttachmentSerializer.loads

`DocumentAttachmentSerializer.loads(json_data: str, *, many: Optional[bool] = None, partial: Optional[Union[bool, Sequence[str], Set[str]]] = None, unknown: Optional[str] = None, **kwargs)`

Same as `load()`, except it takes a JSON string as input.

Parameters

- **json_data** – A JSON string of the data to deserialize.
- **many** – Whether to deserialize *obj* as a collection. If *None*, the value for *self.many* is used.
- **partial** – Whether to ignore missing fields and not require any fields declared. Propagates down to *Nested* fields as well. If its value is an iterable, only missing fields listed in that iterable will be ignored. Use dot delimiters to specify nested fields.
- **unknown** – Whether to exclude, include, or raise an error for unknown fields in the data. Use *EXCLUDE*, *INCLUDE* or *RAISE*. If *None*, the value for *self.unknown* is used.

Returns Deserialized data

New in version 1.0.0.

Changed in version 3.0.0b7: This method returns the deserialized data rather than a (data, errors) tuple. A `ValidationError` is raised if invalid data are passed.

app.serializers.document.DocumentAttachmentSerializer.on_bind_field

`DocumentAttachmentSerializer.on_bind_field`(*field_name*: str, *field_obj*:
marshmallow.fields.Field) → None

Hook to modify a field when it is bound to the *Schema*.

No-op by default.

app.serializers.document.DocumentAttachmentSerializer.process_input

`DocumentAttachmentSerializer.process_input`(*value*, *many*, ***kwargs*)

app.serializers.document.DocumentAttachmentSerializer.validate

`DocumentAttachmentSerializer.validate`(*data*: Union[Mapping[str, Any],
Iterable[Mapping[str, Any]]], *, *many*:
Optional[bool] = None, *partial*:
Optional[Union[bool, Sequence[str], Set[str]]] =
None) → Dict[str, List[str]]

Validate *data* against the schema, returning a dictionary of validation errors.

Parameters

- **data** – The data to validate.
- **many** – Whether to validate *data* as a collection. If *None*, the value for *self.many* is used.
- **partial** – Whether to ignore missing fields and not require any fields declared. Propagates down to *Nested* fields as well. If its value is an iterable, only missing fields listed in that iterable will be ignored. Use dot delimiters to specify nested fields.

Returns A dictionary of validation errors.

New in version 1.1.0.

app.serializers.document.DocumentSerializer

```
class app.serializers.document.DocumentSerializer(*, only: Optional[Union[Sequence[str], Set[str]]]
= None, exclude: Union[Sequence[str], Set[str]] =
(), many: bool = False, context: Optional[Dict] =
None, load_only: Union[Sequence[str], Set[str]] =
(), dump_only: Union[Sequence[str], Set[str]] =
(), partial: Union[bool, Sequence[str], Set[str]] =
False, unknown: Optional[str] = None)
```

Bases: flask_marshmallow.schema.Schema

Attributes

<code>DocumentSerializer.TYPE_MAPPING</code>	
<code>DocumentSerializer.dict_class</code>	
<code>DocumentSerializer.error_messages</code>	Overrides for default schema-level error messages
<code>DocumentSerializer.opts</code>	
<code>DocumentSerializer.set_class</code>	

`app.serializers.document.DocumentSerializer.TYPE_MAPPING`

```
DocumentSerializer.TYPE_MAPPING = {<class 'str'>: <class
'marshmallow.fields.String'>, <class 'bytes'>: <class
'marshmallow.fields.String'>, <class 'datetime.datetime'>: <class
'marshmallow.fields.DateTime'>, <class 'float'>: <class
'marshmallow.fields.Float'>, <class 'bool'>: <class
'marshmallow.fields.Boolean'>, <class 'tuple'>: <class
'marshmallow.fields.Raw'>, <class 'list'>: <class
'marshmallow.fields.Raw'>, <class 'set'>: <class 'marshmallow.fields.Raw'>,
<class 'int'>: <class 'marshmallow.fields.Integer'>, <class 'uuid.UUID'>:
<class 'marshmallow.fields.UUID'>, <class 'datetime.time'>: <class
'marshmallow.fields.Time'>, <class 'datetime.date'>: <class
'marshmallow.fields.Date'>, <class 'datetime.timedelta'>: <class
'marshmallow.fields.TimeDelta'>, <class 'decimal.Decimal'>: <class
'marshmallow.fields.Decimal'>}
```

`app.serializers.document.DocumentSerializer.dict_class`

property `DocumentSerializer.dict_class`: `type`

`app.serializers.document.DocumentSerializer.error_messages`

```
DocumentSerializer.error_messages = {}
    Overrides for default schema-level error messages
```

`app.serializers.document.DocumentSerializer.opts`

```
DocumentSerializer.opts = <marshmallow.schema.SchemaOpts object>
```

app.serializers.document.DocumentSerializer.set_class**property** DocumentSerializer.set_class: **type****Methods**

<code>DocumentSerializer.__init__(*[, only, ...])</code>	
<code>DocumentSerializer.dump(obj, *[, many])</code>	Serialize an object to native Python data types according to this Schema's fields.
<code>DocumentSerializer.dumps(obj, *args[, many])</code>	Same as <code>dump()</code> , except return a JSON-encoded string.
<code>DocumentSerializer.from_dict(fields, *[, name])</code>	Generate a <i>Schema</i> class given a dictionary of fields.
<code>DocumentSerializer.get_attribute(obj, attr, ...)</code>	Defines how to pull values from an object to serialize.
<code>DocumentSerializer.handle_error(error, data, ...)</code>	Custom error handler function for the schema.
<code>DocumentSerializer.jsonify(obj[, many])</code>	Return a JSON response containing the serialized data.
<code>DocumentSerializer.load(data, *[, many, ...])</code>	Deserialize a data structure to an object defined by this Schema's fields.
<code>DocumentSerializer.loads(json_data, *[, ...])</code>	Same as <code>load()</code> , except it takes a JSON string as input.
<code>DocumentSerializer.on_bind_field(field_name, ...)</code>	Hook to modify a field when it is bound to the <i>Schema</i> .
<code>DocumentSerializer.valid_request_file(data)</code>	
<code>DocumentSerializer.validate(data, *[, many, ...])</code>	Validate <i>data</i> against the schema, returning a dictionary of validation errors.
<code>DocumentSerializer.validate_id(document_id)</code>	
<code>DocumentSerializer.wrap(data, **kwargs)</code>	

app.serializers.document.DocumentSerializer.__init__

`DocumentSerializer.__init__(*, only: Optional[Union[Sequence[str], Set[str]]] = None, exclude: Union[Sequence[str], Set[str]] = (), many: bool = False, context: Optional[Dict] = None, load_only: Union[Sequence[str], Set[str]] = (), dump_only: Union[Sequence[str], Set[str]] = (), partial: Union[bool, Sequence[str], Set[str]] = False, unknown: Optional[str] = None)`

app.serializers.document.DocumentSerializer.dump

`DocumentSerializer.dump(obj: Any, *, many: Optional[bool] = None)`

Serialize an object to native Python data types according to this Schema's fields.

Parameters

- **obj** – The object to serialize.
- **many** – Whether to serialize *obj* as a collection. If *None*, the value for *self.many* is used.

Returns Serialized data

New in version 1.0.0.

Changed in version 3.0.0b7: This method returns the serialized data rather than a (data, errors) tuple. A `ValidationError` is raised if *obj* is invalid.

Changed in version 3.0.0rc9: Validation no longer occurs upon serialization.

app.serializers.document.DocumentSerializer.dumps

`DocumentSerializer.dumps(obj: Any, *args, many: Optional[bool] = None, **kwargs)`

Same as `dump()`, except return a JSON-encoded string.

Parameters

- **obj** – The object to serialize.
- **many** – Whether to serialize *obj* as a collection. If *None*, the value for *self.many* is used.

Returns A json string

New in version 1.0.0.

Changed in version 3.0.0b7: This method returns the serialized data rather than a (data, errors) tuple. A `ValidationError` is raised if *obj* is invalid.

app.serializers.document.DocumentSerializer.from_dict

classmethod `DocumentSerializer.from_dict(fields: Dict[str, Union[marshmallow.fields.Field, type]], *, name: str = 'GeneratedSchema') → type`

Generate a *Schema* class given a dictionary of fields.

```
from marshmallow import Schema, fields

PersonSchema = Schema.from_dict({"name": fields.Str()})
print(PersonSchema().load({"name": "David"})) # => {'name': 'David'}
```

Generated schemas are not added to the class registry and therefore cannot be referred to by name in *Nested* fields.

Parameters

- **fields** (*dict*) – Dictionary mapping field names to field instances.
- **name** (*str*) – Optional name for the class, which will appear in the `repr` for the class.

New in version 3.0.0.

app.serializers.document.DocumentSerializer.get_attribute

`DocumentSerializer.get_attribute(obj: Any, attr: str, default: Any)`

Defines how to pull values from an object to serialize.

New in version 2.0.0.

Changed in version 3.0.0a1: Changed position of `obj` and `attr`.

app.serializers.document.DocumentSerializer.handle_error

`DocumentSerializer.handle_error(error: marshmallow.exceptions.ValidationError, data: Any, *, many: bool, **kwargs)`

Custom error handler function for the schema.

Parameters

- **error** – The *ValidationError* raised during (de)serialization.
- **data** – The original input data.
- **many** – Value of `many` on dump or load.
- **partial** – Value of `partial` on load.

New in version 2.0.0.

Changed in version 3.0.0rc9: Receives *many* and *partial* (on deserialization) as keyword arguments.

app.serializers.document.DocumentSerializer.jsonify

`DocumentSerializer.jsonify(obj, many=<object object>, *args, **kwargs)`

Return a JSON response containing the serialized data.

Parameters

- **obj** – Object to serialize.
- **many** (*bool*) – Whether *obj* should be serialized as an instance or as a collection. If unset, defaults to the value of the *many* attribute on this Schema.
- **kwargs** – Additional keyword arguments passed to *flask jsonify*.

Changed in version 0.6.0: Takes the same arguments as *marshmallow.Schema.dump*. Additional keyword arguments are passed to *flask jsonify*.

Changed in version 0.6.3: The *many* argument for this method defaults to the value of the *many* attribute on the Schema. Previously, the *many* argument of this method defaulted to *False*, regardless of the value of *Schema.many*.

app.serializers.document.DocumentSerializer.load

`DocumentSerializer.load(data: Union[Mapping[str, Any], Iterable[Mapping[str, Any]]], *, many: Optional[bool] = None, partial: Optional[Union[bool, Sequence[str], Set[str]]] = None, unknown: Optional[str] = None)`

Deserialize a data structure to an object defined by this Schema's fields.

Parameters

- **data** – The data to deserialize.
- **many** – Whether to deserialize *data* as a collection. If *None*, the value for *self.many* is used.
- **partial** – Whether to ignore missing fields and not require any fields declared. Propagates down to *Nested* fields as well. If its value is an iterable, only missing

fields listed in that iterable will be ignored. Use dot delimiters to specify nested fields.

- **unknown** – Whether to exclude, include, or raise an error for unknown fields in the data. Use *EXCLUDE*, *INCLUDE* or *RAISE*. If *None*, the value for *self.unknown* is used.

Returns Deserialized data

New in version 1.0.0.

Changed in version 3.0.0b7: This method returns the deserialized data rather than a (data, errors) tuple. A `ValidationError` is raised if invalid data are passed.

app.serializers.document.DocumentSerializer.loads

`DocumentSerializer.loads(json_data: str, *, many: Optional[bool] = None, partial: Optional[Union[bool, Sequence[str], Set[str]]] = None, unknown: Optional[str] = None, **kwargs)`

Same as `load()`, except it takes a JSON string as input.

Parameters

- **json_data** – A JSON string of the data to deserialize.
- **many** – Whether to deserialize *obj* as a collection. If *None*, the value for *self.many* is used.
- **partial** – Whether to ignore missing fields and not require any fields declared. Propagates down to *Nested* fields as well. If its value is an iterable, only missing fields listed in that iterable will be ignored. Use dot delimiters to specify nested fields.
- **unknown** – Whether to exclude, include, or raise an error for unknown fields in the data. Use *EXCLUDE*, *INCLUDE* or *RAISE*. If *None*, the value for *self.unknown* is used.

Returns Deserialized data

New in version 1.0.0.

Changed in version 3.0.0b7: This method returns the deserialized data rather than a (data, errors) tuple. A `ValidationError` is raised if invalid data are passed.

app.serializers.document.DocumentSerializer.on_bind_field

`DocumentSerializer.on_bind_field(field_name: str, field_obj: marshmallow.fields.Field) → None`

Hook to modify a field when it is bound to the *Schema*.

No-op by default.

app.serializers.document.DocumentSerializer.valid_request_file

static `DocumentSerializer.valid_request_file(data)`

app.serializers.document.DocumentSerializer.validate

```
DocumentSerializer.validate(data: Union[Mapping[str, Any], Iterable[Mapping[str, Any]]],
                             *, many: Optional[bool] = None, partial:
                             Optional[Union[bool, Sequence[str], Set[str]]] = None) →
                             Dict[str, List[str]]
```

Validate *data* against the schema, returning a dictionary of validation errors.

Parameters

- **data** – The data to validate.
- **many** – Whether to validate *data* as a collection. If *None*, the value for *self.many* is used.
- **partial** – Whether to ignore missing fields and not require any fields declared. Propagates down to **Nested** fields as well. If its value is an iterable, only missing fields listed in that iterable will be ignored. Use dot delimiters to specify nested fields.

Returns A dictionary of validation errors.

New in version 1.1.0.

app.serializers.document.DocumentSerializer.validate_id

```
DocumentSerializer.validate_id(document_id)
```

app.serializers.document.DocumentSerializer.wrap

```
DocumentSerializer.wrap(data, **kwargs)
```

```
class app.serializers.document.DocumentAttachmentSerializer(*, only:
                                                             Optional[Union[Sequence[str],
                                                             Set[str]]] = None, exclude:
                                                             Union[Sequence[str], Set[str]] = (),
                                                             many: bool = False, context:
                                                             Optional[Dict] = None, load_only:
                                                             Union[Sequence[str], Set[str]] = (),
                                                             dump_only: Union[Sequence[str],
                                                             Set[str]] = (), partial: Union[bool,
                                                             Sequence[str], Set[str]] = False,
                                                             unknown: Optional[str] = None)
```

class Meta

Options object for a Schema.

Example usage:

```
class Meta:
    fields = ("id", "email", "date_created")
    exclude = ("password", "secret_attribute")
```

Available options:

- **fields**: Tuple or list of fields to include in the serialized result.
- **additional**: Tuple or list of fields to include *in addition to the* explicitly declared fields. **additional** and **fields** are mutually-exclusive options.

- **include:** Dictionary of additional fields to include in the schema. It is usually better to define fields as class variables, but you may need to use this option, e.g., if your fields are Python keywords. May be an *OrderedDict*.
- **exclude:** Tuple or list of fields to exclude in the serialized result. Nested fields can be represented with dot delimiters.
- **dateformat:** Default format for *Date* <fields.Date> fields.
- **datetimeformat:** Default format for *DateTime* <fields.DateTime> fields.
- **timeformat:** Default format for *Time* <fields.Time> fields.
- **render_module:** Module to use for *loads* <Schema.loads> and *dumps* <Schema.dumps>. Defaults to *json* from the standard library.
- **ordered:** If *True*, order serialization output according to the order in which fields were declared. Output of *Schema.dump* will be a *collections.OrderedDict*.
- **index_errors:** If *True*, errors dictionaries will include the **index** of invalid items in a collection.
- **load_only:** Tuple or list of fields to exclude from serialized results.
- **dump_only:** Tuple or list of fields to exclude from deserialization
- **unknown:** Whether to exclude, include, or raise an error for unknown fields in the data. Use *EXCLUDE*, *INCLUDE* or *RAISE*.
- **register:** Whether to register the *Schema* with marshmallow's internal class registry. Must be *True* if you intend to refer to this *Schema* by class name in *Nested* fields. Only set this to *False* when memory usage is critical. Defaults to *True*.

OPTIONS_CLASS

alias of `marshmallow.schema.SchemaOpts`

```
TYPE_MAPPING = {<class 'str':>: <class 'marshmallow.fields.String'>, <class
'bytes':>: <class 'marshmallow.fields.String'>, <class 'datetime.datetime':>: <class
'marshmallow.fields.DateTime'>, <class 'float':>: <class
'marshmallow.fields.Float'>, <class 'bool':>: <class 'marshmallow.fields.Boolean'>,
<class 'tuple':>: <class 'marshmallow.fields.Raw'>, <class 'list':>: <class
'marshmallow.fields.Raw'>, <class 'set':>: <class 'marshmallow.fields.Raw'>, <class
'int':>: <class 'marshmallow.fields.Integer'>, <class 'uuid.UUID':>: <class
'marshmallow.fields.UUID'>, <class 'datetime.time':>: <class
'marshmallow.fields.Time'>, <class 'datetime.date':>: <class
'marshmallow.fields.Date'>, <class 'datetime.timedelta':>: <class
'marshmallow.fields.TimeDelta'>, <class 'decimal.Decimal':>: <class
'marshmallow.fields.Decimal'>}
```

_bind_field(*field_name*: str, *field_obj*: marshmallow.fields.Field) → None

Bind field to the schema, setting any necessary attributes on the field (e.g. parent and name).

Also set field *load_only* and *dump_only* values if *field_name* was specified in class *Meta*.

static _call_and_store(*getter_func*, *data*, *, *field_name*, *error_store*, *index*=None)

Call *getter_func* with *data* as its argument, and store any *ValidationErrors*.

Parameters

- **getter_func** (callable) – Function for getting the serialized/deserialized value from *data*.
- **data** – The data passed to *getter_func*.
- **field_name** (str) – Field name.

- **index** (*int*) – Index of the item being validated, if validating a collection, otherwise *None*.

```
_declared_fields = {'as_attachment':
    <fields.Integer(dump_default=<marshmallow.missing>, attribute=None,
    validate=<OneOf(choices=[1, 0], labels=[], error='Must be one of: {choices}.')>,
    required=False, load_only=False, dump_only=False,
    load_default=<marshmallow.missing>, allow_none=False, error_messages={'required':
    'Missing data for required field.', 'null': 'Field may not be null.',
    'validator_failed': 'Invalid value.', 'invalid': 'Not a valid integer.',
    'too_large': 'Number too large.'})>>
    _default_error_messages = {'type': 'Invalid input type.', 'unknown': 'Unknown
    field.'}
    _deserialize(data: Union[Mapping[str, Any], Iterable[Mapping[str, Any]]], *, error_store:
        marshmallow.error_store.ErrorStore, many: bool = False, partial=False, unknown='raise',
        index=None) → Union[marshmallow.schema._T, List[marshmallow.schema._T]]
```

Deserialize data.

Parameters

- **data** (*dict*) – The data to deserialize.
- **error_store** (*ErrorStore*) – Structure to store errors.
- **many** (*bool*) – *True* if data should be deserialized as a collection.
- **partial** (*bool | tuple*) – Whether to ignore missing fields and not require any fields declared. Propagates down to Nested fields as well. If its value is an iterable, only missing fields listed in that iterable will be ignored. Use dot delimiters to specify nested fields.
- **unknown** – Whether to exclude, include, or raise an error for unknown fields in the data. Use *EXCLUDE*, *INCLUDE* or *RAISE*.
- **index** (*int*) – Index of the item being serialized (for storing errors) if serializing a collection, otherwise *None*.

Returns A dictionary of the deserialized data.

```
_do_load(data: Union[Mapping[str, Any], Iterable[Mapping[str, Any]]], *, many: Optional[bool] = None,
    partial: Optional[Union[bool, Sequence[str], Set[str]]] = None, unknown: Optional[str] = None,
    postprocess: bool = True)
```

Deserialize *data*, returning the deserialized result. This method is private API.

Parameters

- **data** – The data to deserialize.
- **many** – Whether to deserialize *data* as a collection. If *None*, the value for *self.many* is used.
- **partial** – Whether to validate required fields. If its value is an iterable, only fields listed in that iterable will be ignored will be allowed missing. If *True*, all fields will be allowed missing. If *None*, the value for *self.partial* is used.
- **unknown** – Whether to exclude, include, or raise an error for unknown fields in the data. Use *EXCLUDE*, *INCLUDE* or *RAISE*. If *None*, the value for *self.unknown* is used.
- **postprocess** – Whether to run *post_load* methods..

Returns Deserialized data

```
_has_processors(tag) → bool
```

```
_hooks = {('pre_load', False): ['process_input']}
```


_init_fields() → None

Update self.fields, self.load_fields, and self.dump_fields based on schema options. This method is private API.

_invoke_dump_processors(tag: str, data, *, many: bool, original_data=None)

_invoke_field_validators(*, error_store: marshmallow.error_store.ErrorStore, data, many: bool)

_invoke_load_processors(tag: str, data, *, many: bool, original_data, partial: Union[bool, Sequence[str], Set[str]])

_invoke_processors(tag: str, *, pass_many: bool, data, many: bool, original_data=None, **kwargs)

_invoke_schema_validators(*, error_store: marshmallow.error_store.ErrorStore, pass_many: bool, data, original_data, many: bool, partial: Union[bool, Sequence[str], Set[str]], field_errors: bool = False)

_normalize_nested_options() → None

Apply then flatten nested schema options. This method is private API.

_run_validator(validator_func, output, *, original_data, error_store, many, partial, pass_original, index=None)

_serialize(obj: Union[marshmallow.schema._T, Iterable[marshmallow.schema._T]], *, many: bool = False)

Serialize obj.

Parameters

- **obj** – The object(s) to serialize.
- **many** (bool) – True if data should be serialized as a collection.

Returns A dictionary of the serialized data

Changed in version 1.0.0: Renamed from marshal.

property dict_class: type

dump(obj: Any, *, many: Optional[bool] = None)

Serialize an object to native Python data types according to this Schema's fields.

Parameters

- **obj** – The object to serialize.
- **many** – Whether to serialize obj as a collection. If None, the value for self.many is used.

Returns Serialized data

New in version 1.0.0.

Changed in version 3.0.0b7: This method returns the serialized data rather than a (data, errors) tuple. A ValidationError is raised if obj is invalid.

Changed in version 3.0.0rc9: Validation no longer occurs upon serialization.

dumps(obj: Any, *args, many: Optional[bool] = None, **kwargs)

Same as [dump\(\)](#), except return a JSON-encoded string.

Parameters

- **obj** – The object to serialize.
- **many** – Whether to serialize obj as a collection. If None, the value for self.many is used.

Returns A json string

New in version 1.0.0.

Changed in version 3.0.0b7: This method returns the serialized data rather than a (data, errors) tuple. A `ValidationError` is raised if obj is invalid.

error_messages = {}

Overrides for default schema-level error messages

fields

Dictionary mapping field_names -> Field objects

classmethod from_dict(fields: Dict[str, Union[marshmallow.fields.Field, type]], *, name: str = 'GeneratedSchema') → type

Generate a *Schema* class given a dictionary of fields.

```
from marshmallow import Schema, fields

PersonSchema = Schema.from_dict({"name": fields.Str()})
print(PersonSchema().load({"name": "David"})) # => {'name': 'David'}
```

Generated schemas are not added to the class registry and therefore cannot be referred to by name in *Nested* fields.

Parameters

- **fields** (dict) – Dictionary mapping field names to field instances.
- **name** (str) – Optional name for the class, which will appear in the repr for the class.

New in version 3.0.0.

get_attribute(obj: Any, attr: str, default: Any)

Defines how to pull values from an object to serialize.

New in version 2.0.0.

Changed in version 3.0.0a1: Changed position of obj and attr.

handle_error(error: marshmallow.exceptions.ValidationError, data: Any, *, many: bool, **kwargs)

Custom error handler function for the schema.

Parameters

- **error** – The *ValidationError* raised during (de)serialization.
- **data** – The original input data.
- **many** – Value of many on dump or load.
- **partial** – Value of partial on load.

New in version 2.0.0.

Changed in version 3.0.0rc9: Receives *many* and *partial* (on deserialization) as keyword arguments.

jsonify(obj, many=<object object>, *args, **kwargs)

Return a JSON response containing the serialized data.

Parameters

- **obj** – Object to serialize.
- **many** (bool) – Whether *obj* should be serialized as an instance or as a collection. If unset, defaults to the value of the *many* attribute on this Schema.
- **kwargs** – Additional keyword arguments passed to *flask.jsonify*.

Changed in version 0.6.0: Takes the same arguments as *marshmallow.Schema.dump*. Additional keyword arguments are passed to *flask.jsonify*.

Changed in version 0.6.3: The *many* argument for this method defaults to the value of the *many* attribute on the Schema. Previously, the *many* argument of this method defaulted to False, regardless of the value of *Schema.many*.

load(data: Union[Mapping[str, Any], Iterable[Mapping[str, Any]]], *, many: Optional[bool] = None, partial: Optional[Union[bool, Sequence[str], Set[str]]] = None, unknown: Optional[str] = None)
 Deserialize a data structure to an object defined by this Schema's fields.

Parameters

- **data** – The data to deserialize.
- **many** – Whether to deserialize *data* as a collection. If *None*, the value for *self.many* is used.
- **partial** – Whether to ignore missing fields and not require any fields declared. Propagates down to *Nested* fields as well. If its value is an iterable, only missing fields listed in that iterable will be ignored. Use dot delimiters to specify nested fields.
- **unknown** – Whether to exclude, include, or raise an error for unknown fields in the data. Use *EXCLUDE*, *INCLUDE* or *RAISE*. If *None*, the value for *self.unknown* is used.

Returns Deserialized data

New in version 1.0.0.

Changed in version 3.0.0b7: This method returns the deserialized data rather than a (data, errors) tuple. A *ValidationError* is raised if invalid data are passed.

loads(json_data: str, *, many: Optional[bool] = None, partial: Optional[Union[bool, Sequence[str], Set[str]]] = None, unknown: Optional[str] = None, **kwargs)
 Same as *load()*, except it takes a JSON string as input.

Parameters

- **json_data** – A JSON string of the data to deserialize.
- **many** – Whether to deserialize *obj* as a collection. If *None*, the value for *self.many* is used.
- **partial** – Whether to ignore missing fields and not require any fields declared. Propagates down to *Nested* fields as well. If its value is an iterable, only missing fields listed in that iterable will be ignored. Use dot delimiters to specify nested fields.
- **unknown** – Whether to exclude, include, or raise an error for unknown fields in the data. Use *EXCLUDE*, *INCLUDE* or *RAISE*. If *None*, the value for *self.unknown* is used.

Returns Deserialized data

New in version 1.0.0.

Changed in version 3.0.0b7: This method returns the deserialized data rather than a (data, errors) tuple. A *ValidationError* is raised if invalid data are passed.

on_bind_field(field_name: str, field_obj: marshmallow.fields.Field) → None
 Hook to modify a field when it is bound to the *Schema*.

No-op by default.

opts = <marshmallow.schema.SchemaOpts object>

process_input(value, many, **kwargs)

property set_class: type

validate(*data*: Union[Mapping[str, Any], Iterable[Mapping[str, Any]]], *, *many*: Optional[bool] = None, *partial*: Optional[Union[bool, Sequence[str], Set[str]]] = None) → Dict[str, List[str]]

Validate *data* against the schema, returning a dictionary of validation errors.

Parameters

- **data** – The data to validate.
- **many** – Whether to validate *data* as a collection. If *None*, the value for *self.many* is used.
- **partial** – Whether to ignore missing fields and not require any fields declared. Propagates down to *Nested* fields as well. If its value is an iterable, only missing fields listed in that iterable will be ignored. Use dot delimiters to specify nested fields.

Returns A dictionary of validation errors.

New in version 1.1.0.

```
class app.serializers.document.DocumentSerializer(*, only: Optional[Union[Sequence[str], Set[str]]]
                                                = None, exclude: Union[Sequence[str], Set[str]] =
                                                (), many: bool = False, context: Optional[Dict] =
                                                None, load_only: Union[Sequence[str], Set[str]] =
                                                (), dump_only: Union[Sequence[str], Set[str]] = (),
                                                partial: Union[bool, Sequence[str], Set[str]] =
                                                False, unknown: Optional[str] = None)
```

class Meta

ordered = True

OPTIONS_CLASS

alias of marshmallow.schema.SchemaOpts

```
TYPE_MAPPING = {<class 'str':>: <class 'marshmallow.fields.String'>, <class
'bytes':>: <class 'marshmallow.fields.String'>, <class 'datetime.datetime':>: <class
'marshmallow.fields.DateTime'>, <class 'float':>: <class
'marshmallow.fields.Float'>, <class 'bool':>: <class 'marshmallow.fields.Boolean'>,
<class 'tuple':>: <class 'marshmallow.fields.Raw'>, <class 'list':>: <class
'marshmallow.fields.Raw'>, <class 'set':>: <class 'marshmallow.fields.Raw'>, <class
'int':>: <class 'marshmallow.fields.Integer'>, <class 'uuid.UUID':>: <class
'marshmallow.fields.UUID'>, <class 'datetime.time':>: <class
'marshmallow.fields.Time'>, <class 'datetime.date':>: <class
'marshmallow.fields.Date'>, <class 'datetime.timedelta':>: <class
'marshmallow.fields.TimeDelta'>, <class 'decimal.Decimal':>: <class
'marshmallow.fields.Decimal'>}
```

_bind_field(*field_name*: str, *field_obj*: marshmallow.fields.Field) → None

Bind field to the schema, setting any necessary attributes on the field (e.g. parent and name).

Also set field *load_only* and *dump_only* values if *field_name* was specified in class *Meta*.

static _call_and_store(*getter_func*, *data*, *, *field_name*, *error_store*, *index*=None)

Call *getter_func* with *data* as its argument, and store any *ValidationErrors*.

Parameters

- **getter_func** (*callable*) – Function for getting the serialized/deserialized value from *data*.
- **data** – The data passed to *getter_func*.

- **field_name** (*str*) – Field name.
- **index** (*int*) – Index of the item being validated, if validating a collection, otherwise *None*.

```

_declared_fields = {'created_at':
<fields.TimestampField(dump_default=<marshmallow.missing>, attribute=None,
validate=None, required=False, load_only=False, dump_only=True,
load_default=<marshmallow.missing>, allow_none=False, error_messages={'required':
'Missing data for required field.', 'null': 'Field may not be null.',
'validator_failed': 'Invalid value.'})>, 'created_by':
<fields.Nested(dump_default=<marshmallow.missing>, attribute=None, validate=None,
required=False, load_only=False, dump_only=False,
load_default=<marshmallow.missing>, allow_none=False, error_messages={'required':
'Missing data for required field.', 'null': 'Field may not be null.',
'validator_failed': 'Invalid value.', 'type': 'Invalid type.'})>, 'deleted_at':
<fields.TimestampField(dump_default=<marshmallow.missing>, attribute=None,
validate=None, required=False, load_only=False, dump_only=True,
load_default=<marshmallow.missing>, allow_none=False, error_messages={'required':
'Missing data for required field.', 'null': 'Field may not be null.',
'validator_failed': 'Invalid value.'})>, 'directory_path':
<fields.String(dump_default=<marshmallow.missing>, attribute=None, validate=None,
required=False, load_only=True, dump_only=False, load_default=<marshmallow.missing>,
allow_none=False, error_messages={'required': 'Missing data for required field.',
'null': 'Field may not be null.', 'validator_failed': 'Invalid value.', 'invalid':
'Not a valid string.', 'invalid_utf8': 'Not a valid utf-8 string.'})>, 'id':
<fields.Integer(dump_default=<marshmallow.missing>, attribute=None, validate=None,
required=False, load_only=False, dump_only=False,
load_default=<marshmallow.missing>, allow_none=False, error_messages={'required':
'Missing data for required field.', 'null': 'Field may not be null.',
'validator_failed': 'Invalid value.', 'invalid': 'Not a valid integer.',
'too_large': 'Number too large.'})>, 'internal_filename':
<fields.String(dump_default=<marshmallow.missing>, attribute=None, validate=None,
required=False, load_only=False, dump_only=False,
load_default=<marshmallow.missing>, allow_none=False, error_messages={'required':
'Missing data for required field.', 'null': 'Field may not be null.',
'validator_failed': 'Invalid value.', 'invalid': 'Not a valid string.',
'invalid_utf8': 'Not a valid utf-8 string.'})>, 'mime_type':
<fields.String(dump_default=<marshmallow.missing>, attribute=None, validate=None,
required=False, load_only=False, dump_only=False,
load_default=<marshmallow.missing>, allow_none=False, error_messages={'required':
'Missing data for required field.', 'null': 'Field may not be null.',
'validator_failed': 'Invalid value.', 'invalid': 'Not a valid string.',
'invalid_utf8': 'Not a valid utf-8 string.'})>, 'name':
<fields.String(dump_default=<marshmallow.missing>, attribute=None, validate=None,
required=False, load_only=False, dump_only=False,
load_default=<marshmallow.missing>, allow_none=False, error_messages={'required':
'Missing data for required field.', 'null': 'Field may not be null.',
'validator_failed': 'Invalid value.', 'invalid': 'Not a valid string.',
'invalid_utf8': 'Not a valid utf-8 string.'})>, 'size':
<fields.Integer(dump_default=<marshmallow.missing>, attribute=None, validate=None,
required=False, load_only=False, dump_only=False,
load_default=<marshmallow.missing>, allow_none=False, error_messages={'required':
'Missing data for required field.', 'null': 'Field may not be null.',
'validator_failed': 'Invalid value.', 'invalid': 'Not a valid integer.',
'too_large': 'Number too large.'})>, 'updated_at':
<fields.TimestampField(dump_default=<marshmallow.missing>, attribute=None,
validate=None, required=False, load_only=False, dump_only=True,
load_default=<marshmallow.missing>, allow_none=False, error_messages={'required':
'Missing data for required field.', 'null': 'Field may not be null.',
'validator_failed': 'Invalid value.'})>>

```

```
_default_error_messages = {'type': 'Invalid input type.', 'unknown': 'Unknown field.'}
```

```
_deserialize(data: Union[Mapping[str, Any], Iterable[Mapping[str, Any]]], *, error_store:
    marshmallow.error_store.ErrorStore, many: bool = False, partial=False, unknown='raise',
    index=None) → Union[marshmallow.schema._T, List[marshmallow.schema._T]]
```

Deserialize data.

Parameters

- **data** (*dict*) – The data to deserialize.
- **error_store** (*ErrorStore*) – Structure to store errors.
- **many** (*bool*) – *True* if data should be deserialized as a collection.
- **partial** (*bool | tuple*) – Whether to ignore missing fields and not require any fields declared. Propagates down to *Nested* fields as well. If its value is an iterable, only missing fields listed in that iterable will be ignored. Use dot delimiters to specify nested fields.
- **unknown** – Whether to exclude, include, or raise an error for unknown fields in the data. Use *EXCLUDE*, *INCLUDE* or *RAISE*.
- **index** (*int*) – Index of the item being serialized (for storing errors) if serializing a collection, otherwise *None*.

Returns A dictionary of the deserialized data.

```
_do_load(data: Union[Mapping[str, Any], Iterable[Mapping[str, Any]]], *, many: Optional[bool] = None,
    partial: Optional[Union[bool, Sequence[str], Set[str]]] = None, unknown: Optional[str] = None,
    postprocess: bool = True)
```

Deserialize *data*, returning the deserialized result. This method is private API.

Parameters

- **data** – The data to deserialize.
- **many** – Whether to deserialize *data* as a collection. If *None*, the value for *self.many* is used.
- **partial** – Whether to validate required fields. If its value is an iterable, only fields listed in that iterable will be ignored will be allowed missing. If *True*, all fields will be allowed missing. If *None*, the value for *self.partial* is used.
- **unknown** – Whether to exclude, include, or raise an error for unknown fields in the data. Use *EXCLUDE*, *INCLUDE* or *RAISE*. If *None*, the value for *self.unknown* is used.
- **postprocess** – Whether to run *post_load* methods..

Returns Deserialized data

```
_has_processors(tag) → bool
```

```
_hooks = defaultdict(<class 'list'>, {'validates': ['validate_id'], ('post_dump', False): ['wrap']})
```

```
_init_fields() → None
```

Update *self.fields*, *self.load_fields*, and *self.dump_fields* based on schema options. This method is private API.

```
_invoke_dump_processors(tag: str, data, *, many: bool, original_data=None)
```

```
_invoke_field_validators(*, error_store: marshmallow.error_store.ErrorStore, data, many: bool)
```

```
_invoke_load_processors(tag: str, data, *, many: bool, original_data, partial: Union[bool, Sequence[str], Set[str]])
```

_invoke_processors(tag: str, *, pass_many: bool, data, many: bool, original_data=None, **kwargs)
_invoke_schema_validators(*, error_store: marshmallow.error_store.ErrorStore, pass_many: bool, data, original_data, many: bool, partial: Union[bool, Sequence[str], Set[str]], field_errors: bool = False)

_normalize_nested_options() → None

Apply then flatten nested schema options. This method is private API.

_run_validator(validator_func, output, *, original_data, error_store, many, partial, pass_original, index=None)

_serialize(obj: Union[marshmallow.schema._T, Iterable[marshmallow.schema._T]], *, many: bool = False)

Serialize obj.

Parameters

- **obj** – The object(s) to serialize.
- **many** (bool) – True if data should be serialized as a collection.

Returns A dictionary of the serialized data

Changed in version 1.0.0: Renamed from `marshal`.

property dict_class: type

dump(obj: Any, *, many: Optional[bool] = None)

Serialize an object to native Python data types according to this Schema's fields.

Parameters

- **obj** – The object to serialize.
- **many** – Whether to serialize *obj* as a collection. If *None*, the value for *self.many* is used.

Returns Serialized data

New in version 1.0.0.

Changed in version 3.0.0b7: This method returns the serialized data rather than a (data, errors) tuple. A `ValidationError` is raised if *obj* is invalid.

Changed in version 3.0.0rc9: Validation no longer occurs upon serialization.

dumps(obj: Any, *args, many: Optional[bool] = None, **kwargs)

Same as `dump()`, except return a JSON-encoded string.

Parameters

- **obj** – The object to serialize.
- **many** – Whether to serialize *obj* as a collection. If *None*, the value for *self.many* is used.

Returns A json string

New in version 1.0.0.

Changed in version 3.0.0b7: This method returns the serialized data rather than a (data, errors) tuple. A `ValidationError` is raised if *obj* is invalid.

error_messages = {}

Overrides for default schema-level error messages

fields

Dictionary mapping field_names -> Field objects

classmethod from_dict(*fields: Dict[str, Union[marshmallow.fields.Field, type]], *, name: str = 'GeneratedSchema'*) → type
Generate a *Schema* class given a dictionary of fields.

```
from marshmallow import Schema, fields

PersonSchema = Schema.from_dict({"name": fields.Str()})
print(PersonSchema().load({"name": "David"})) # => {'name': 'David'}
```

Generated schemas are not added to the class registry and therefore cannot be referred to by name in *Nested* fields.

Parameters

- **fields** (*dict*) – Dictionary mapping field names to field instances.
- **name** (*str*) – Optional name for the class, which will appear in the `repr` for the class.

New in version 3.0.0.

get_attribute(*obj: Any, attr: str, default: Any*)
Defines how to pull values from an object to serialize.

New in version 2.0.0.

Changed in version 3.0.0a1: Changed position of `obj` and `attr`.

handle_error(*error: marshmallow.exceptions.ValidationError, data: Any, *, many: bool, **kwargs*)
Custom error handler function for the schema.

Parameters

- **error** – The *ValidationError* raised during (de)serialization.
- **data** – The original input data.
- **many** – Value of `many` on dump or load.
- **partial** – Value of `partial` on load.

New in version 2.0.0.

Changed in version 3.0.0rc9: Receives *many* and *partial* (on deserialization) as keyword arguments.

jsonify(*obj, many=<object object>, *args, **kwargs*)
Return a JSON response containing the serialized data.

Parameters

- **obj** – Object to serialize.
- **many** (*bool*) – Whether *obj* should be serialized as an instance or as a collection. If unset, defaults to the value of the *many* attribute on this Schema.
- **kwargs** – Additional keyword arguments passed to *flask.jsonify*.

Changed in version 0.6.0: Takes the same arguments as *marshmallow.Schema.dump*. Additional keyword arguments are passed to *flask.jsonify*.

Changed in version 0.6.3: The *many* argument for this method defaults to the value of the *many* attribute on the Schema. Previously, the *many* argument of this method defaulted to `False`, regardless of the value of *Schema.many*.

load(*data: Union[Mapping[str, Any], Iterable[Mapping[str, Any]]], *, many: Optional[bool] = None, partial: Optional[Union[bool, Sequence[str], Set[str]]] = None, unknown: Optional[str] = None*)
Deserialize a data structure to an object defined by this Schema's fields.

Parameters

- **data** – The data to deserialize.
- **many** – Whether to deserialize *data* as a collection. If *None*, the value for *self.many* is used.
- **partial** – Whether to ignore missing fields and not require any fields declared. Propagates down to *Nested* fields as well. If its value is an iterable, only missing fields listed in that iterable will be ignored. Use dot delimiters to specify nested fields.
- **unknown** – Whether to exclude, include, or raise an error for unknown fields in the data. Use *EXCLUDE*, *INCLUDE* or *RAISE*. If *None*, the value for *self.unknown* is used.

Returns Deserialized data

New in version 1.0.0.

Changed in version 3.0.0b7: This method returns the deserialized data rather than a (data, errors) tuple. A *ValidationError* is raised if invalid data are passed.

loads(*json_data*: str, *, *many*: Optional[bool] = None, *partial*: Optional[Union[bool, Sequence[str], Set[str]]] = None, *unknown*: Optional[str] = None, **kwargs)

Same as *load()*, except it takes a JSON string as input.

Parameters

- **json_data** – A JSON string of the data to deserialize.
- **many** – Whether to deserialize *obj* as a collection. If *None*, the value for *self.many* is used.
- **partial** – Whether to ignore missing fields and not require any fields declared. Propagates down to *Nested* fields as well. If its value is an iterable, only missing fields listed in that iterable will be ignored. Use dot delimiters to specify nested fields.
- **unknown** – Whether to exclude, include, or raise an error for unknown fields in the data. Use *EXCLUDE*, *INCLUDE* or *RAISE*. If *None*, the value for *self.unknown* is used.

Returns Deserialized data

New in version 1.0.0.

Changed in version 3.0.0b7: This method returns the deserialized data rather than a (data, errors) tuple. A *ValidationError* is raised if invalid data are passed.

on_bind_field(*field_name*: str, *field_obj*: *marshmallow.fields.Field*) → None

Hook to modify a field when it is bound to the *Schema*.

No-op by default.

opts = <marshmallow.schema.SchemaOpts object>

property set_class: type

static valid_request_file(*data*)

validate(*data*: Union[Mapping[str, Any], Iterable[Mapping[str, Any]]], *, *many*: Optional[bool] = None, *partial*: Optional[Union[bool, Sequence[str], Set[str]]] = None) → Dict[str, List[str]]

Validate *data* against the schema, returning a dictionary of validation errors.

Parameters

- **data** – The data to validate.
- **many** – Whether to validate *data* as a collection. If *None*, the value for *self.many* is used.

- **partial** – Whether to ignore missing fields and not require any fields declared. Propagates down to `Nested` fields as well. If its value is an iterable, only missing fields listed in that iterable will be ignored. Use dot delimiters to specify nested fields.

Returns A dictionary of validation errors.

New in version 1.1.0.

validate_id(*document_id*)

wrap(*data*, ***kwargs*)

app.serializers.role

Description

Classes

RoleName(*, load_default, missing, ...)

RoleSerializer(*[, only, exclude, many, ...])

app.serializers.role.RoleName

```
class app.serializers.role.RoleName(*, load_default: Any = <marshmallow.missing>, missing: Any =
    <marshmallow.missing>, dump_default: Any =
    <marshmallow.missing>, default: Any = <marshmallow.missing>,
    data_key: Optional[str] = None, attribute: Optional[str] = None,
    validate: Optional[Union[Callable[[Any], Any],
    Iterable[Callable[[Any], Any]]]] = None, required: bool = False,
    allow_none: Optional[bool] = None, load_only: bool = False,
    dump_only: bool = False, error_messages: Optional[Dict[str, str]] =
    None, metadata: Optional[Mapping[str, Any]] = None,
    **additional_metadata)
```

Bases: `marshmallow.fields.Field`

Attributes

<i>RoleName.context</i>	The context dictionary for the parent Schema.
<i>RoleName.default</i>	
<i>RoleName.default_error_messages</i>	Default error messages for various kinds of errors.
<i>RoleName.missing</i>	
<i>RoleName.name</i>	
<i>RoleName.parent</i>	

continues on next page

Table 114 – continued from previous page

RoleName.root

app.serializers.role.RoleName.context**property RoleName.context**

The context dictionary for the parent Schema.

app.serializers.role.RoleName.default**property RoleName.default****app.serializers.role.RoleName.default_error_messages**

```
RoleName.default_error_messages = {'null': 'Field may not be null.',  
'required': 'Missing data for required field.', 'validator_failed':  
'Invalid value.'}
```

Default error messages for various kinds of errors. The keys in this dictionary are passed to *Field.make_error*. The values are error messages passed to *marshmallow.exceptions.ValidationError*.

app.serializers.role.RoleName.missing**property RoleName.missing****app.serializers.role.RoleName.name**

```
RoleName.name = None
```

app.serializers.role.RoleName.parent

```
RoleName.parent = None
```

app.serializers.role.RoleName.root

```
RoleName.root = None
```

Methods

<code>RoleName.__init__(*[, load_default, ...])</code>	
<code>RoleName.deserialize(value[, attr, data])</code>	Deserialize value.
<code>RoleName.fail(key, **kwargs)</code>	Helper method that raises a <i>ValidationError</i> with an error message from <code>self.error_messages</code> .
<code>RoleName.get_value(obj, attr[, accessor, ...])</code>	Return the value for a given key from an object.
<code>RoleName.make_error(key, **kwargs)</code>	Helper method to make a <i>ValidationError</i> with an error message from <code>self.error_messages</code> .
<code>RoleName.serialize(attr, obj[, accessor])</code>	Pulls the value for the given key from the object, applies the field's formatting and returns the result.

app.serializers.role.RoleName.__init__

`RoleName.__init__(*, load_default: Any = <marshmallow.missing>, missing: Any = <marshmallow.missing>, dump_default: Any = <marshmallow.missing>, default: Any = <marshmallow.missing>, data_key: Optional[str] = None, attribute: Optional[str] = None, validate: Optional[Union[Callable[[Any], Any], Iterable[Callable[[Any], Any]]]] = None, required: bool = False, allow_none: Optional[bool] = None, load_only: bool = False, dump_only: bool = False, error_messages: Optional[Dict[str, str]] = None, metadata: Optional[Mapping[str, Any]] = None, **additional_metadata) → None`

app.serializers.role.RoleName.deserialize

`RoleName.deserialize(value: Any, attr: Optional[str] = None, data: Optional[Mapping[str, Any]] = None, **kwargs)`

Deserialize value.

Parameters

- **value** – The value to deserialize.
- **attr** – The attribute/key in `data` to deserialize.
- **data** – The raw input data passed to `Schema.load`.
- **kwargs** – Field-specific keyword arguments.

Raises **ValidationError** – If an invalid value is passed or if a required value is missing.

app.serializers.role.RoleName.fail

`RoleName.fail(key: str, **kwargs)`

Helper method that raises a *ValidationError* with an error message from `self.error_messages`.

Deprecated since version 3.0.0: Use `make_error` <*marshmallow.fields.Field.make_error*> instead.

app.serializers.role.RoleName.get_value

`RoleName.get_value(obj, attr, accessor=None, default=<marshmallow.missing>)`

Return the value for a given key from an object.

Parameters

- **obj** (*object*) – The object to get the value from.
- **attr** (*str*) – The attribute/key in *obj* to get the value from.
- **accessor** (*callable*) – A callable used to retrieve the value of *attr* from the object *obj*. Defaults to *marshmallow.utils.get_value*.

app.serializers.role.RoleName.make_error

`RoleName.make_error(key: str, **kwargs) → marshmallow.exceptions.ValidationError`

Helper method to make a *ValidationError* with an error message from `self.error_messages`.

app.serializers.role.RoleName.serialize

`RoleName.serialize(attr: str, obj: Any, accessor: Optional[Callable[[Any, str, Any], Any]] = None, **kwargs)`

Pulls the value for the given key from the object, applies the field's formatting and returns the result.

Parameters

- **attr** – The attribute/key to get from the object.
- **obj** – The object to access the attribute/key from.
- **accessor** – Function used to access values from *obj*.
- **kwargs** – Field-specific keyword arguments.

app.serializers.role.RoleSerializer

```
class app.serializers.role.RoleSerializer(*, only: Optional[Union[Sequence[str], Set[str]]] = None,
                                         exclude: Union[Sequence[str], Set[str]] = (), many: bool =
                                         False, context: Optional[Dict] = None, load_only:
                                         Union[Sequence[str], Set[str]] = (), dump_only:
                                         Union[Sequence[str], Set[str]] = (), partial: Union[bool,
                                         Sequence[str], Set[str]] = False, unknown: Optional[str] =
                                         None)
```

Bases: `flask_marshmallow.schema.Schema`

Attributes

<code>RoleSerializer.TYPE_MAPPING</code>	
<code>RoleSerializer.dict_class</code>	
<code>RoleSerializer.error_messages</code>	Overrides for default schema-level error messages
<code>RoleSerializer.opts</code>	
<code>RoleSerializer.set_class</code>	

`app.serializers.role.RoleSerializer.TYPE_MAPPING`

```
RoleSerializer.TYPE_MAPPING = {<class 'str'>: <class
'marshmallow.fields.String'>, <class 'bytes'>: <class
'marshmallow.fields.String'>, <class 'datetime.datetime'>: <class
'marshmallow.fields.DateTime'>, <class 'float'>: <class
'marshmallow.fields.Float'>, <class 'bool'>: <class
'marshmallow.fields.Boolean'>, <class 'tuple'>: <class
'marshmallow.fields.Raw'>, <class 'list'>: <class
'marshmallow.fields.Raw'>, <class 'set'>: <class 'marshmallow.fields.Raw'>,
<class 'int'>: <class 'marshmallow.fields.Integer'>, <class 'uuid.UUID'>:
<class 'marshmallow.fields.UUID'>, <class 'datetime.time'>: <class
'marshmallow.fields.Time'>, <class 'datetime.date'>: <class
'marshmallow.fields.Date'>, <class 'datetime.timedelta'>: <class
'marshmallow.fields.TimeDelta'>, <class 'decimal.Decimal'>: <class
'marshmallow.fields.Decimal'>}
```

`app.serializers.role.RoleSerializer.dict_class`

```
property RoleSerializer.dict_class: type
```

`app.serializers.role.RoleSerializer.error_messages`

```
RoleSerializer.error_messages = {}
    Overrides for default schema-level error messages
```

`app.serializers.role.RoleSerializer.opts`

```
RoleSerializer.opts = <marshmallow.schema.SchemaOpts object>
```

app.serializers.role.RoleSerializer.set_class**property** RoleSerializer.set_class: **type****Methods**

<code>RoleSerializer.__init__(*[, only, exclude, ...])</code>	
<code>RoleSerializer.dump(obj, *[, many])</code>	Serialize an object to native Python data types according to this Schema's fields.
<code>RoleSerializer.dumps(obj, *args[, many])</code>	Same as <code>dump()</code> , except return a JSON-encoded string.
<code>RoleSerializer.from_dict(fields, *[, name])</code>	Generate a <i>Schema</i> class given a dictionary of fields.
<code>RoleSerializer.get_attribute(obj, attr, default)</code>	Defines how to pull values from an object to serialize.
<code>RoleSerializer.handle_error(error, data, *, ...)</code>	Custom error handler function for the schema.
<code>RoleSerializer.jsonify(obj[, many])</code>	Return a JSON response containing the serialized data.
<code>RoleSerializer.load(data, *[, many, ...])</code>	Deserialize a data structure to an object defined by this Schema's fields.
<code>RoleSerializer.loads(json_data, *[, many, ...])</code>	Same as <code>load()</code> , except it takes a JSON string as input.
<code>RoleSerializer.on_bind_field(field_name, ...)</code>	Hook to modify a field when it is bound to the <i>Schema</i> .
<code>RoleSerializer.sluglify_name(item, many, ...)</code>	
<code>RoleSerializer.validate(data, *[, many, partial])</code>	Validate <i>data</i> against the schema, returning a dictionary of validation errors.
<code>RoleSerializer.validate_id(role_id)</code>	
<code>RoleSerializer.validate_name(value, **kwargs)</code>	

app.serializers.role.RoleSerializer.__init__

`RoleSerializer.__init__(*, only: Optional[Union[Sequence[str], Set[str]]] = None, exclude: Union[Sequence[str], Set[str]] = (), many: bool = False, context: Optional[Dict] = None, load_only: Union[Sequence[str], Set[str]] = (), dump_only: Union[Sequence[str], Set[str]] = (), partial: Union[bool, Sequence[str], Set[str]] = False, unknown: Optional[str] = None)`

app.serializers.role.RoleSerializer.dump

`RoleSerializer.dump(obj: Any, *, many: Optional[bool] = None)`

Serialize an object to native Python data types according to this Schema's fields.

Parameters

- **obj** – The object to serialize.
- **many** – Whether to serialize *obj* as a collection. If *None*, the value for *self.many* is used.

Returns Serialized data

New in version 1.0.0.

Changed in version 3.0.0b7: This method returns the serialized data rather than a (data, errors) tuple. A `ValidationError` is raised if *obj* is invalid.

Changed in version 3.0.0rc9: Validation no longer occurs upon serialization.

app.serializers.role.RoleSerializer.dumps

`RoleSerializer.dumps(obj: Any, *args, many: Optional[bool] = None, **kwargs)`

Same as `dump()`, except return a JSON-encoded string.

Parameters

- **obj** – The object to serialize.
- **many** – Whether to serialize *obj* as a collection. If *None*, the value for *self.many* is used.

Returns A json string

New in version 1.0.0.

Changed in version 3.0.0b7: This method returns the serialized data rather than a (data, errors) tuple. A `ValidationError` is raised if *obj* is invalid.

app.serializers.role.RoleSerializer.from_dict

classmethod `RoleSerializer.from_dict(fields: Dict[str, Union[marshmallow.fields.Field, type]], *, name: str = 'GeneratedSchema') → type`

Generate a *Schema* class given a dictionary of fields.

```
from marshmallow import Schema, fields

PersonSchema = Schema.from_dict({"name": fields.Str()})
print(PersonSchema().load({"name": "David"}))  # => {'name': 'David'}
```

Generated schemas are not added to the class registry and therefore cannot be referred to by name in *Nested* fields.

Parameters

- **fields** (*dict*) – Dictionary mapping field names to field instances.
- **name** (*str*) – Optional name for the class, which will appear in the `repr` for the class.

New in version 3.0.0.

`app.serializers.role.RoleSerializer.get_attribute`

`RoleSerializer.get_attribute(obj: Any, attr: str, default: Any)`

Defines how to pull values from an object to serialize.

New in version 2.0.0.

Changed in version 3.0.0a1: Changed position of `obj` and `attr`.

`app.serializers.role.RoleSerializer.handle_error`

`RoleSerializer.handle_error(error: marshmallow.exceptions.ValidationError, data: Any, *, many: bool, **kwargs)`

Custom error handler function for the schema.

Parameters

- **error** – The *ValidationError* raised during (de)serialization.
- **data** – The original input data.
- **many** – Value of `many` on dump or load.
- **partial** – Value of `partial` on load.

New in version 2.0.0.

Changed in version 3.0.0rc9: Receives *many* and *partial* (on deserialization) as keyword arguments.

`app.serializers.role.RoleSerializer.jsonify`

`RoleSerializer.jsonify(obj, many=<object object>, *args, **kwargs)`

Return a JSON response containing the serialized data.

Parameters

- **obj** – Object to serialize.
- **many** (*bool*) – Whether *obj* should be serialized as an instance or as a collection. If unset, defaults to the value of the *many* attribute on this Schema.
- **kwargs** – Additional keyword arguments passed to *flask.jsonify*.

Changed in version 0.6.0: Takes the same arguments as *marshmallow.Schema.dump*. Additional keyword arguments are passed to *flask.jsonify*.

Changed in version 0.6.3: The *many* argument for this method defaults to the value of the *many* attribute on the Schema. Previously, the *many* argument of this method defaulted to `False`, regardless of the value of *Schema.many*.

`app.serializers.role.RoleSerializer.load`

`RoleSerializer.load(data: Union[Mapping[str, Any], Iterable[Mapping[str, Any]]], *, many: Optional[bool] = None, partial: Optional[Union[bool, Sequence[str], Set[str]]] = None, unknown: Optional[str] = None)`

Deserialize a data structure to an object defined by this Schema's fields.

Parameters

- **data** – The data to deserialize.
- **many** – Whether to deserialize *data* as a collection. If *None*, the value for *self.many* is used.
- **partial** – Whether to ignore missing fields and not require any fields declared. Propagates down to *Nested* fields as well. If its value is an iterable, only missing

fields listed in that iterable will be ignored. Use dot delimiters to specify nested fields.

- **unknown** – Whether to exclude, include, or raise an error for unknown fields in the data. Use *EXCLUDE*, *INCLUDE* or *RAISE*. If *None*, the value for *self.unknown* is used.

Returns Deserialized data

New in version 1.0.0.

Changed in version 3.0.0b7: This method returns the deserialized data rather than a (data, errors) tuple. A `ValidationError` is raised if invalid data are passed.

app.serializers.role.RoleSerializer.loads

`RoleSerializer.loads(json_data: str, *, many: Optional[bool] = None, partial: Optional[Union[bool, Sequence[str], Set[str]]] = None, unknown: Optional[str] = None, **kwargs)`

Same as `load()`, except it takes a JSON string as input.

Parameters

- **json_data** – A JSON string of the data to deserialize.
- **many** – Whether to deserialize *obj* as a collection. If *None*, the value for *self.many* is used.
- **partial** – Whether to ignore missing fields and not require any fields declared. Propagates down to *Nested* fields as well. If its value is an iterable, only missing fields listed in that iterable will be ignored. Use dot delimiters to specify nested fields.
- **unknown** – Whether to exclude, include, or raise an error for unknown fields in the data. Use *EXCLUDE*, *INCLUDE* or *RAISE*. If *None*, the value for *self.unknown* is used.

Returns Deserialized data

New in version 1.0.0.

Changed in version 3.0.0b7: This method returns the deserialized data rather than a (data, errors) tuple. A `ValidationError` is raised if invalid data are passed.

app.serializers.role.RoleSerializer.on_bind_field

`RoleSerializer.on_bind_field(field_name: str, field_obj: marshmallow.fields.Field) → None`

Hook to modify a field when it is bound to the *Schema*.

No-op by default.

app.serializers.role.RoleSerializer.sluglify_name

`RoleSerializer.sluglify_name(item, many, **kwargs)`

app.serializers.role.RoleSerializer.validate

`RoleSerializer.validate(data: Union[Mapping[str, Any], Iterable[Mapping[str, Any]]], *, many: Optional[bool] = None, partial: Optional[Union[bool, Sequence[str], Set[str]]] = None) → Dict[str, List[str]]`

Validate *data* against the schema, returning a dictionary of validation errors.

Parameters

- **data** – The data to validate.
- **many** – Whether to validate *data* as a collection. If *None*, the value for *self.many* is used.
- **partial** – Whether to ignore missing fields and not require any fields declared. Propagates down to *Nested* fields as well. If its value is an iterable, only missing fields listed in that iterable will be ignored. Use dot delimiters to specify nested fields.

Returns A dictionary of validation errors.

New in version 1.1.0.

app.serializers.role.RoleSerializer.validate_id

`RoleSerializer.validate_id(role_id)`

app.serializers.role.RoleSerializer.validate_name

`RoleSerializer.validate_name(value, **kwargs)`

```
class app.serializers.role.RoleName(*, load_default: Any = <marshmallow.missing>, missing: Any =
    <marshmallow.missing>, dump_default: Any =
    <marshmallow.missing>, default: Any = <marshmallow.missing>,
    data_key: Optional[str] = None, attribute: Optional[str] = None,
    validate: Optional[Union[Callable[[Any], Any],
    Iterable[Callable[[Any], Any]]]] = None, required: bool = False,
    allow_none: Optional[bool] = None, load_only: bool = False,
    dump_only: bool = False, error_messages: Optional[Dict[str, str]] =
    None, metadata: Optional[Mapping[str, Any]] = None,
    **additional_metadata)
```

`_CHECK_ATTRIBUTE = True`

`_bind_to_schema(field_name, schema)`

Update field with values from its parent schema. Called by `Schema._bind_field`.

Parameters

- **field_name** (*str*) – Field name set in schema.
- **schema** (*Schema* / *Field*) – Parent object.

`_creation_index = 129`

`_deserialize(value, *args, **kwargs)`

Deserialize value. Concrete *Field* classes should implement this method.

Parameters

- **value** – The value to be deserialized.

- **attr** – The attribute/key in *data* to be deserialized.
- **data** – The raw input data passed to the *Schema.load*.
- **kwargs** – Field-specific keyword arguments.

Raises **ValidationError** – In case of formatting or validation failure.

Returns The deserialized value.

Changed in version 2.0.0: Added **attr** and **data** parameters.

Changed in version 3.0.0: Added ****kwargs** to signature.

_serialize(*value*, **args*, ***kwargs*)

Serializes *value* to a basic Python datatype. Noop by default. Concrete Field classes should implement this method.

Example:

```
class TitleCase(Field):
    def _serialize(self, value, attr, obj, **kwargs):
        if not value:
            return ''
        return str(value).title()
```

Parameters

- **value** – The value to be serialized.
- **attr** (*str*) – The attribute or key on the object to be serialized.
- **obj** (*object*) – The object the value was pulled from.
- **kwargs** (*dict*) – Field-specific keyword arguments.

Returns The serialized value

_validate(*value*)

Perform validation on *value*. Raise a **ValidationError** if validation does not succeed.

property **_validate_all**

_validate_missing(*value*)

Validate missing values. Raise a **ValidationError** if *value* should be considered missing.

property **context**

The context dictionary for the parent Schema.

property **default**

default_error_messages = {'null': 'Field may not be null.', 'required': 'Missing data for required field.', 'validator_failed': 'Invalid value.'}

Default error messages for various kinds of errors. The keys in this dictionary are passed to *Field.make_error*. The values are error messages passed to *marshmallow.exceptions.ValidationError*.

deserialize(*value*: Any, *attr*: Optional[str] = None, *data*: Optional[Mapping[str, Any]] = None, ***kwargs*)

Deserialize value.

Parameters

- **value** – The value to deserialize.
- **attr** – The attribute/key in *data* to deserialize.

- **data** – The raw input data passed to *Schema.load*.
- **kwargs** – Field-specific keyword arguments.

Raises `ValidationError` – If an invalid value is passed or if a required value is missing.

fail(*key: str, **kwargs*)

Helper method that raises a *ValidationError* with an error message from *self.error_messages*.

Deprecated since version 3.0.0: Use *make_error* <*marshmallow.fields.Field.make_error*> instead.

get_value(*obj, attr, accessor=None, default=<marshmallow.missing>*)

Return the value for a given key from an object.

Parameters

- **obj** (*object*) – The object to get the value from.
- **attr** (*str*) – The attribute/key in *obj* to get the value from.
- **accessor** (*callable*) – A callable used to retrieve the value of *attr* from the object *obj*. Defaults to *marshmallow.utils.get_value*.

make_error(*key: str, **kwargs*) → *marshmallow.exceptions.ValidationError*

Helper method to make a *ValidationError* with an error message from *self.error_messages*.

property missing

name = *None*

parent = *None*

root = *None*

serialize(*attr: str, obj: Any, accessor: Optional[Callable[[Any, str, Any], Any]] = None, **kwargs*)

Pulls the value for the given key from the object, applies the field's formatting and returns the result.

Parameters

- **attr** – The attribute/key to get from the object.
- **obj** – The object to access the attribute/key from.
- **accessor** – Function used to access values from *obj*.
- **kwargs** – Field-specific keyword arguments.

```
class app.serializers.role.RoleSerializer(*, only: Optional[Union[Sequence[str], Set[str]]] = None,
                                         exclude: Union[Sequence[str], Set[str]] = (), many: bool =
                                         False, context: Optional[Dict] = None, load_only:
                                         Union[Sequence[str], Set[str]] = (), dump_only:
                                         Union[Sequence[str], Set[str]] = (), partial: Union[bool,
                                         Sequence[str], Set[str]] = False, unknown: Optional[str] =
                                         None)
```

class Meta

ordered = *True*

OPTIONS_CLASS

alias of *marshmallow.schema.SchemaOpts*

```

TYPE_MAPPING = {<class 'str':>: <class 'marshmallow.fields.String'>, <class
'bytes':>: <class 'marshmallow.fields.String'>, <class 'datetime.datetime':>: <class
'marshmallow.fields.DateTime'>, <class 'float':>: <class
'marshmallow.fields.Float'>, <class 'bool':>: <class 'marshmallow.fields.Boolean'>,
<class 'tuple':>: <class 'marshmallow.fields.Raw'>, <class 'list':>: <class
'marshmallow.fields.Raw'>, <class 'set':>: <class 'marshmallow.fields.Raw'>, <class
'int':>: <class 'marshmallow.fields.Integer'>, <class 'uuid.UUID':>: <class
'marshmallow.fields.UUID'>, <class 'datetime.time':>: <class
'marshmallow.fields.Time'>, <class 'datetime.date':>: <class
'marshmallow.fields.Date'>, <class 'datetime.timedelta':>: <class
'marshmallow.fields.TimeDelta'>, <class 'decimal.Decimal':>: <class
'marshmallow.fields.Decimal'>}

```

`_bind_field`(*field_name*: str, *field_obj*: marshmallow.fields.Field) → None

Bind field to the schema, setting any necessary attributes on the field (e.g. parent and name).

Also set field `load_only` and `dump_only` values if `field_name` was specified in class `Meta`.

`static _call_and_store`(*getter_func*, *data*, *, *field_name*, *error_store*, *index*=None)

Call `getter_func` with `data` as its argument, and store any `ValidationErrors`.

Parameters

- **`getter_func`** (*callable*) – Function for getting the serialized/deserialized value from data.
- **`data`** – The data passed to `getter_func`.
- **`field_name`** (*str*) – Field name.
- **`index`** (*int*) – Index of the item being validated, if validating a collection, otherwise *None*.

```

_declared_fields = {'created_at':
<fields.TimestampField(dump_default=<marshmallow.missing>, attribute=None,
validate=None, required=False, load_only=False, dump_only=True,
load_default=<marshmallow.missing>, allow_none=False, error_messages={'required':
'Missing data for required field.', 'null': 'Field may not be null.',
'validator_failed': 'Invalid value.'})>, 'deleted_at':
<fields.TimestampField(dump_default=<marshmallow.missing>, attribute=None,
validate=None, required=False, load_only=False, dump_only=True,
load_default=<marshmallow.missing>, allow_none=False, error_messages={'required':
'Missing data for required field.', 'null': 'Field may not be null.',
'validator_failed': 'Invalid value.'})>, 'description':
<fields.String(dump_default=<marshmallow.missing>, attribute=None, validate=None,
required=False, load_only=False, dump_only=False,
load_default=<marshmallow.missing>, allow_none=False, error_messages={'required':
'Missing data for required field.', 'null': 'Field may not be null.',
'validator_failed': 'Invalid value.', 'invalid': 'Not a valid string.',
'invalid_utf8': 'Not a valid utf-8 string.'})>, 'id':
<fields.Integer(dump_default=<marshmallow.missing>, attribute=None, validate=None,
required=False, load_only=False, dump_only=False,
load_default=<marshmallow.missing>, allow_none=False, error_messages={'required':
'Missing data for required field.', 'null': 'Field may not be null.',
'validator_failed': 'Invalid value.', 'invalid': 'Not a valid integer.',
'too_large': 'Number too large.'})>, 'label':
<fields.String(dump_default=<marshmallow.missing>, attribute=None, validate=None,
required=False, load_only=False, dump_only=False,
load_default=<marshmallow.missing>, allow_none=False, error_messages={'required':
'Missing data for required field.', 'null': 'Field may not be null.',
'validator_failed': 'Invalid value.', 'invalid': 'Not a valid string.',
'invalid_utf8': 'Not a valid utf-8 string.'})>, 'name':
<fields.String(dump_default=<marshmallow.missing>, attribute=None, validate=None,
required=False, load_only=False, dump_only=True, load_default=<marshmallow.missing>,
allow_none=False, error_messages={'required': 'Missing data for required field.',
'null': 'Field may not be null.', 'validator_failed': 'Invalid value.', 'invalid':
'Not a valid string.', 'invalid_utf8': 'Not a valid utf-8 string.'})>,
'updated_at': <fields.TimestampField(dump_default=<marshmallow.missing>,
attribute=None, validate=None, required=False, load_only=False, dump_only=True,
load_default=<marshmallow.missing>, allow_none=False, error_messages={'required':
'Missing data for required field.', 'null': 'Field may not be null.',
'validator_failed': 'Invalid value.'})>}]

_default_error_messages = {'type': 'Invalid input type.', 'unknown': 'Unknown
field.'}

```

```

_deserialize(data: Union[Mapping[str, Any], Iterable[Mapping[str, Any]]], *, error_store:
marshmallow.error_store.ErrorStore, many: bool = False, partial=False, unknown='raise',
index=None) → Union[marshmallow.schema._T, List[marshmallow.schema._T]]

```

Deserialize data.

Parameters

- **data** (*dict*) – The data to deserialize.
- **error_store** (*ErrorStore*) – Structure to store errors.
- **many** (*bool*) – *True* if data should be deserialized as a collection.
- **partial** (*bool* / *tuple*) – Whether to ignore missing fields and not require any fields de-

clared. Propagates down to `Nested` fields as well. If its value is an iterable, only missing fields listed in that iterable will be ignored. Use dot delimiters to specify nested fields.

- **unknown** – Whether to exclude, include, or raise an error for unknown fields in the data. Use `EXCLUDE`, `INCLUDE` or `RAISE`.
- **index** (*int*) – Index of the item being serialized (for storing errors) if serializing a collection, otherwise `None`.

Returns A dictionary of the deserialized data.

```
_do_load(data: Union[Mapping[str, Any], Iterable[Mapping[str, Any]]], *, many: Optional[bool] = None,
          partial: Optional[Union[bool, Sequence[str], Set[str]]] = None, unknown: Optional[str] = None,
          postprocess: bool = True)
```

Deserialize *data*, returning the deserialized result. This method is private API.

Parameters

- **data** – The data to deserialize.
- **many** – Whether to deserialize *data* as a collection. If `None`, the value for *self.many* is used.
- **partial** – Whether to validate required fields. If its value is an iterable, only fields listed in that iterable will be ignored will be allowed missing. If `True`, all fields will be allowed missing. If `None`, the value for *self.partial* is used.
- **unknown** – Whether to exclude, include, or raise an error for unknown fields in the data. Use `EXCLUDE`, `INCLUDE` or `RAISE`. If `None`, the value for *self.unknown* is used.
- **postprocess** – Whether to run `post_load` methods..

Returns Deserialized data

```
_has_processors(tag) → bool
```

```
_hooks = defaultdict(<class 'list'>, {('post_load', False): ['sluglify_name'],
'validates': ['validate_id', 'validate_name']})
```

```
_init_fields() → None
```

Update *self.fields*, *self.load_fields*, and *self.dump_fields* based on schema options. This method is private API.

```
_invoke_dump_processors(tag: str, data, *, many: bool, original_data=None)
```

```
_invoke_field_validators(*, error_store: marshmallow.error_store.ErrorStore, data, many: bool)
```

```
_invoke_load_processors(tag: str, data, *, many: bool, original_data, partial: Union[bool,
Sequence[str], Set[str]])
```

```
_invoke_processors(tag: str, *, pass_many: bool, data, many: bool, original_data=None, **kwargs)
```

```
_invoke_schema_validators(*, error_store: marshmallow.error_store.ErrorStore, pass_many: bool, data,
original_data, many: bool, partial: Union[bool, Sequence[str], Set[str]],
field_errors: bool = False)
```

```
_normalize_nested_options() → None
```

Apply then flatten nested schema options. This method is private API.

```
_run_validator(validator_func, output, *, original_data, error_store, many, partial, pass_original,
index=None)
```

```
_serialize(obj: Union[marshmallow.schema._T, Iterable[marshmallow.schema._T]], *, many: bool =
False)
```

Serialize *obj*.

Parameters

- **obj** – The object(s) to serialize.
- **many** (*bool*) – *True* if data should be serialized as a collection.

Returns A dictionary of the serialized data

Changed in version 1.0.0: Renamed from `marshal`.

property dict_class: **type**

dump(*obj: Any, *, many: Optional[bool] = None*)

Serialize an object to native Python data types according to this Schema's fields.

Parameters

- **obj** – The object to serialize.
- **many** – Whether to serialize *obj* as a collection. If *None*, the value for *self.many* is used.

Returns Serialized data

New in version 1.0.0.

Changed in version 3.0.0b7: This method returns the serialized data rather than a (`data`, `errors`) tuple. A `ValidationError` is raised if *obj* is invalid.

Changed in version 3.0.0rc9: Validation no longer occurs upon serialization.

dumps(*obj: Any, *args, many: Optional[bool] = None, **kwargs*)

Same as `dump()`, except return a JSON-encoded string.

Parameters

- **obj** – The object to serialize.
- **many** – Whether to serialize *obj* as a collection. If *None*, the value for *self.many* is used.

Returns A json string

New in version 1.0.0.

Changed in version 3.0.0b7: This method returns the serialized data rather than a (`data`, `errors`) tuple. A `ValidationError` is raised if *obj* is invalid.

error_messages = {}

Overrides for default schema-level error messages

fields

Dictionary mapping field_names -> Field objects

classmethod from_dict(*fields: Dict[str, Union[marshmallow.fields.Field, type]], *, name: str = 'GeneratedSchema'*) → type

Generate a *Schema* class given a dictionary of fields.

```
from marshmallow import Schema, fields

PersonSchema = Schema.from_dict({"name": fields.Str()})
print(PersonSchema().load({"name": "David"})) # => {'name': 'David'}
```

Generated schemas are not added to the class registry and therefore cannot be referred to by name in *Nested* fields.

Parameters

- **fields** (*dict*) – Dictionary mapping field names to field instances.

- **name** (*str*) – Optional name for the class, which will appear in the `repr` for the class.

New in version 3.0.0.

get_attribute(*obj: Any, attr: str, default: Any*)

Defines how to pull values from an object to serialize.

New in version 2.0.0.

Changed in version 3.0.0a1: Changed position of `obj` and `attr`.

handle_error(*error: marshmallow.exceptions.ValidationError, data: Any, *, many: bool, **kwargs*)

Custom error handler function for the schema.

Parameters

- **error** – The *ValidationError* raised during (de)serialization.
- **data** – The original input data.
- **many** – Value of `many` on dump or load.
- **partial** – Value of `partial` on load.

New in version 2.0.0.

Changed in version 3.0.0rc9: Receives *many* and *partial* (on deserialization) as keyword arguments.

jsonify(*obj, many=<object object>, *args, **kwargs*)

Return a JSON response containing the serialized data.

Parameters

- **obj** – Object to serialize.
- **many** (*bool*) – Whether *obj* should be serialized as an instance or as a collection. If unset, defaults to the value of the *many* attribute on this Schema.
- **kwargs** – Additional keyword arguments passed to *flask jsonify*.

Changed in version 0.6.0: Takes the same arguments as *marshmallow.Schema.dump*. Additional keyword arguments are passed to *flask jsonify*.

Changed in version 0.6.3: The *many* argument for this method defaults to the value of the *many* attribute on the Schema. Previously, the *many* argument of this method defaulted to `False`, regardless of the value of *Schema.many*.

load(*data: Union[Mapping[str, Any], Iterable[Mapping[str, Any]]], *, many: Optional[bool] = None, partial: Optional[Union[bool, Sequence[str], Set[str]]] = None, unknown: Optional[str] = None*)

Deserialize a data structure to an object defined by this Schema's fields.

Parameters

- **data** – The data to deserialize.
- **many** – Whether to deserialize *data* as a collection. If *None*, the value for *self.many* is used.
- **partial** – Whether to ignore missing fields and not require any fields declared. Propagates down to *Nested* fields as well. If its value is an iterable, only missing fields listed in that iterable will be ignored. Use dot delimiters to specify nested fields.
- **unknown** – Whether to exclude, include, or raise an error for unknown fields in the data. Use *EXCLUDE*, *INCLUDE* or *RAISE*. If *None*, the value for *self.unknown* is used.

Returns Deserialized data

New in version 1.0.0.

Changed in version 3.0.0b7: This method returns the deserialized data rather than a (data, errors) tuple. A `ValidationError` is raised if invalid data are passed.

loads(*json_data*: str, *, *many*: Optional[bool] = None, *partial*: Optional[Union[bool, Sequence[str], Set[str]]] = None, *unknown*: Optional[str] = None, **kwargs)

Same as `load()`, except it takes a JSON string as input.

Parameters

- **json_data** – A JSON string of the data to deserialize.
- **many** – Whether to deserialize *obj* as a collection. If *None*, the value for *self.many* is used.
- **partial** – Whether to ignore missing fields and not require any fields declared. Propagates down to *Nested* fields as well. If its value is an iterable, only missing fields listed in that iterable will be ignored. Use dot delimiters to specify nested fields.
- **unknown** – Whether to exclude, include, or raise an error for unknown fields in the data. Use *EXCLUDE*, *INCLUDE* or *RAISE*. If *None*, the value for *self.unknown* is used.

Returns Deserialized data

New in version 1.0.0.

Changed in version 3.0.0b7: This method returns the deserialized data rather than a (data, errors) tuple. A `ValidationError` is raised if invalid data are passed.

on_bind_field(*field_name*: str, *field_obj*: *marshmallow.fields.Field*) → None

Hook to modify a field when it is bound to the *Schema*.

No-op by default.

opts = <marshmallow.schema.SchemaOpts object>

property set_class: type

sluglify_name(*item*, *many*, **kwargs)

validate(*data*: Union[Mapping[str, Any], Iterable[Mapping[str, Any]]], *, *many*: Optional[bool] = None, *partial*: Optional[Union[bool, Sequence[str], Set[str]]] = None) → Dict[str, List[str]]

Validate *data* against the schema, returning a dictionary of validation errors.

Parameters

- **data** – The data to validate.
- **many** – Whether to validate *data* as a collection. If *None*, the value for *self.many* is used.
- **partial** – Whether to ignore missing fields and not require any fields declared. Propagates down to *Nested* fields as well. If its value is an iterable, only missing fields listed in that iterable will be ignored. Use dot delimiters to specify nested fields.

Returns A dictionary of validation errors.

New in version 1.1.0.

validate_id(*role_id*)

validate_name(*value*, **kwargs)

app.serializers.user

Description

Classes

<i>UserExportWordSerializer</i> (*[, only, exclude, ...])
<i>UserSerializer</i> (*[, only, exclude, many, ...])
<i>VerifyRoleId</i> (*[, load_default, missing, ...])

app.serializers.user.UserExportWordSerializer

class app.serializers.user.UserExportWordSerializer(*, only: Optional[Union[Sequence[str], Set[str]]] = None, exclude: Union[Sequence[str], Set[str]] = (), many: bool = False, context: Optional[Dict] = None, load_only: Union[Sequence[str], Set[str]] = (), dump_only: Union[Sequence[str], Set[str]] = (), partial: Union[bool, Sequence[str], Set[str]] = False, unknown: Optional[str] = None)

Bases: flask_marshmallow.schema.Schema

Attributes

<i>UserExportWordSerializer</i> . TYPE_MAPPING	
<i>UserExportWordSerializer</i> .dict_class	
<i>UserExportWordSerializer</i> . error_messages	Overrides for default schema-level error messages
<i>UserExportWordSerializer</i> .opts	
<i>UserExportWordSerializer</i> .set_class	

app.serializers.user.UserExportWordSerializer.TYPE_MAPPING

```
UserExportWordSerializer.TYPE_MAPPING = {<class 'str'>: <class
'marshmallow.fields.String'>, <class 'bytes'>: <class
'marshmallow.fields.String'>, <class 'datetime.datetime'>: <class
'marshmallow.fields.DateTime'>, <class 'float'>: <class
'marshmallow.fields.Float'>, <class 'bool'>: <class
'marshmallow.fields.Boolean'>, <class 'tuple'>: <class
'marshmallow.fields.Raw'>, <class 'list'>: <class
'marshmallow.fields.Raw'>, <class 'set'>: <class 'marshmallow.fields.Raw'>,
<class 'int'>: <class 'marshmallow.fields.Integer'>, <class 'uuid.UUID'>:
<class 'marshmallow.fields.UUID'>, <class 'datetime.time'>: <class
'marshmallow.fields.Time'>, <class 'datetime.date'>: <class
'marshmallow.fields.Date'>, <class 'datetime.timedelta'>: <class
'marshmallow.fields.TimeDelta'>, <class 'decimal.Decimal'>: <class
'marshmallow.fields.Decimal'>}
```

app.serializers.user.UserExportWordSerializer.dict_class

property UserExportWordSerializer.dict_class: type

app.serializers.user.UserExportWordSerializer.error_messages

UserExportWordSerializer.error_messages = {}
 Overrides for default schema-level error messages

app.serializers.user.UserExportWordSerializer.opts

UserExportWordSerializer.opts = <marshmallow.schema.SchemaOpts object>

app.serializers.user.UserExportWordSerializer.set_class

property UserExportWordSerializer.set_class: type

Methods

<code>UserExportWordSerializer.__init__(*[, only, ...])</code>	
<code>UserExportWordSerializer.dump(obj, *[, many])</code>	Serialize an object to native Python data types according to this Schema's fields.
<code>UserExportWordSerializer.dumps(obj, *args[, ...])</code>	Same as <code>dump()</code> , except return a JSON-encoded string.
<code>UserExportWordSerializer.from_dict(fields, *)</code>	Generate a <i>Schema</i> class given a dictionary of fields.
<code>UserExportWordSerializer.get_attribute(obj, ...)</code>	Defines how to pull values from an object to serialize.

continues on next page

Table 120 – continued from previous page

<code>UserExportWordSerializer.handle_error(error, ...)</code>	Custom error handler function for the schema.
<code>UserExportWordSerializer.jsonify(obj[, many])</code>	Return a JSON response containing the serialized data.
<code>UserExportWordSerializer.load(data, *, ...)</code>	Deserialize a data structure to an object defined by this Schema's fields.
<code>UserExportWordSerializer.loads(json_data, *)</code>	Same as <code>load()</code> , except it takes a JSON string as input.
<code>UserExportWordSerializer.on_bind_field(...)</code>	Hook to modify a field when it is bound to the <i>Schema</i> .
<code>UserExportWordSerializer.process_input(...)</code>	
<code>UserExportWordSerializer.validate(data, *, ...)</code>	Validate <i>data</i> against the schema, returning a dictionary of validation errors.

app.serializers.user.UserExportWordSerializer.__init__

`UserExportWordSerializer.__init__(*, only: Optional[Union[Sequence[str], Set[str]]] = None, exclude: Union[Sequence[str], Set[str]] = (), many: bool = False, context: Optional[Dict] = None, load_only: Union[Sequence[str], Set[str]] = (), dump_only: Union[Sequence[str], Set[str]] = (), partial: Union[bool, Sequence[str], Set[str]] = False, unknown: Optional[str] = None)`

app.serializers.user.UserExportWordSerializer.dump

`UserExportWordSerializer.dump(obj: Any, *, many: Optional[bool] = None)`

Serialize an object to native Python data types according to this Schema's fields.

Parameters

- **obj** – The object to serialize.
- **many** – Whether to serialize *obj* as a collection. If *None*, the value for *self.many* is used.

Returns Serialized data

New in version 1.0.0.

Changed in version 3.0.0b7: This method returns the serialized data rather than a (data, errors) tuple. A `ValidationError` is raised if *obj* is invalid.

Changed in version 3.0.0rc9: Validation no longer occurs upon serialization.

app.serializers.user.UserExportWordSerializer.dumps

`UserExportWordSerializer.dumps(obj: Any, *args, many: Optional[bool] = None, **kwargs)`
Same as `dump()`, except return a JSON-encoded string.

Parameters

- **obj** – The object to serialize.
- **many** – Whether to serialize *obj* as a collection. If *None*, the value for *self.many* is used.

Returns A json string

New in version 1.0.0.

Changed in version 3.0.0b7: This method returns the serialized data rather than a (data, errors) tuple. A `ValidationError` is raised if *obj* is invalid.

app.serializers.user.UserExportWordSerializer.from_dict

classmethod `UserExportWordSerializer.from_dict(fields: Dict[str, Union[marshmallow.fields.Field, type]], *, name: str = 'GeneratedSchema') → type`

Generate a *Schema* class given a dictionary of fields.

```
from marshmallow import Schema, fields

PersonSchema = Schema.from_dict({"name": fields.Str()})
print(PersonSchema().load({"name": "David"})) # => {'name': 'David'}
```

Generated schemas are not added to the class registry and therefore cannot be referred to by name in *Nested* fields.

Parameters

- **fields** (*dict*) – Dictionary mapping field names to field instances.
- **name** (*str*) – Optional name for the class, which will appear in the repr for the class.

New in version 3.0.0.

app.serializers.user.UserExportWordSerializer.get_attribute

`UserExportWordSerializer.get_attribute(obj: Any, attr: str, default: Any)`

Defines how to pull values from an object to serialize.

New in version 2.0.0.

Changed in version 3.0.0a1: Changed position of *obj* and *attr*.

app.serializers.user.UserExportWordSerializer.handle_error

`UserExportWordSerializer.handle_error`(*error: marshmallow.exceptions.ValidationError*,
data: Any, *, *many: bool*, ***kwargs*)

Custom error handler function for the schema.

Parameters

- **error** – The *ValidationError* raised during (de)serialization.
- **data** – The original input data.
- **many** – Value of *many* on dump or load.
- **partial** – Value of *partial* on load.

New in version 2.0.0.

Changed in version 3.0.0rc9: Receives *many* and *partial* (on deserialization) as keyword arguments.

app.serializers.user.UserExportWordSerializer.jsonify

`UserExportWordSerializer.jsonify`(*obj*, *many=<object object>*, **args*, ***kwargs*)

Return a JSON response containing the serialized data.

Parameters

- **obj** – Object to serialize.
- **many** (*bool*) – Whether *obj* should be serialized as an instance or as a collection. If unset, defaults to the value of the *many* attribute on this Schema.
- **kwargs** – Additional keyword arguments passed to *flask.jsonify*.

Changed in version 0.6.0: Takes the same arguments as *marshmallow.Schema.dump*. Additional keyword arguments are passed to *flask.jsonify*.

Changed in version 0.6.3: The *many* argument for this method defaults to the value of the *many* attribute on the Schema. Previously, the *many* argument of this method defaulted to False, regardless of the value of *Schema.many*.

app.serializers.user.UserExportWordSerializer.load

`UserExportWordSerializer.load`(*data: Union[Mapping[str, Any], Iterable[Mapping[str, Any]]]*, *, *many: Optional[bool] = None*, *partial: Optional[Union[bool, Sequence[str], Set[str]]] = None*,
unknown: Optional[str] = None)

Deserialize a data structure to an object defined by this Schema's fields.

Parameters

- **data** – The data to deserialize.
- **many** – Whether to deserialize *data* as a collection. If *None*, the value for *self.many* is used.
- **partial** – Whether to ignore missing fields and not require any fields declared. Propagates down to *Nested* fields as well. If its value is an iterable, only missing fields listed in that iterable will be ignored. Use dot delimiters to specify nested fields.
- **unknown** – Whether to exclude, include, or raise an error for unknown fields in the data. Use *EXCLUDE*, *INCLUDE* or *RAISE*. If *None*, the value for *self.unknown* is used.

Returns Deserialized data

New in version 1.0.0.

Changed in version 3.0.0b7: This method returns the deserialized data rather than a (data, errors) tuple. A `ValidationError` is raised if invalid data are passed.

`app.serializers.user.UserExportWordSerializer.loads`

`UserExportWordSerializer.loads(json_data: str, *, many: Optional[bool] = None, partial: Optional[Union[bool, Sequence[str], Set[str]]] = None, unknown: Optional[str] = None, **kwargs)`

Same as `load()`, except it takes a JSON string as input.

Parameters

- **json_data** – A JSON string of the data to deserialize.
- **many** – Whether to deserialize *obj* as a collection. If *None*, the value for *self.many* is used.
- **partial** – Whether to ignore missing fields and not require any fields declared. Propagates down to *Nested* fields as well. If its value is an iterable, only missing fields listed in that iterable will be ignored. Use dot delimiters to specify nested fields.
- **unknown** – Whether to exclude, include, or raise an error for unknown fields in the data. Use *EXCLUDE*, *INCLUDE* or *RAISE*. If *None*, the value for *self.unknown* is used.

Returns Deserialized data

New in version 1.0.0.

Changed in version 3.0.0b7: This method returns the deserialized data rather than a (data, errors) tuple. A `ValidationError` is raised if invalid data are passed.

`app.serializers.user.UserExportWordSerializer.on_bind_field`

`UserExportWordSerializer.on_bind_field(field_name: str, field_obj: marshmallow.fields.Field) → None`

Hook to modify a field when it is bound to the *Schema*.

No-op by default.

`app.serializers.user.UserExportWordSerializer.process_input`

`UserExportWordSerializer.process_input(value, many, **kwargs)`

`app.serializers.user.UserExportWordSerializer.validate`

`UserExportWordSerializer.validate(data: Union[Mapping[str, Any], Iterable[Mapping[str, Any]]], *, many: Optional[bool] = None, partial: Optional[Union[bool, Sequence[str], Set[str]]] = None) → Dict[str, List[str]]`

Validate *data* against the schema, returning a dictionary of validation errors.

Parameters

- **data** – The data to validate.
- **many** – Whether to validate *data* as a collection. If *None*, the value for *self.many* is used.

- **partial** – Whether to ignore missing fields and not require any fields declared. Propagates down to `Nested` fields as well. If its value is an iterable, only missing fields listed in that iterable will be ignored. Use dot delimiters to specify nested fields.

Returns A dictionary of validation errors.

New in version 1.1.0.

app.serializers.user.UserSerializer

```
class app.serializers.user.UserSerializer(*, only: Optional[Union[Sequence[str], Set[str]]] = None,
                                         exclude: Union[Sequence[str], Set[str]] = (), many: bool =
                                         False, context: Optional[Dict] = None, load_only:
                                         Union[Sequence[str], Set[str]] = (), dump_only:
                                         Union[Sequence[str], Set[str]] = (), partial: Union[bool,
                                         Sequence[str], Set[str]] = False, unknown: Optional[str] =
                                         None)
```

Bases: flask_marshmallow.schema.Schema

Attributes

`UserSerializer.TYPE_MAPPING`

`UserSerializer.dict_class`

`UserSerializer.error_messages` Overrides for default schema-level error messages

`UserSerializer.opts`

`UserSerializer.set_class`

app.serializers.user.UserSerializer.TYPE_MAPPING

```
UserSerializer.TYPE_MAPPING = {<class 'str':>: <class
'marshmallow.fields.String'>, <class 'bytes':>: <class
'marshmallow.fields.String'>, <class 'datetime.datetime':>: <class
'marshmallow.fields.DateTime'>, <class 'float':>: <class
'marshmallow.fields.Float'>, <class 'bool':>: <class
'marshmallow.fields.Boolean'>, <class 'tuple':>: <class
'marshmallow.fields.Raw'>, <class 'list':>: <class
'marshmallow.fields.Raw'>, <class 'set':>: <class 'marshmallow.fields.Raw'>,
<class 'int':>: <class 'marshmallow.fields.Integer'>, <class 'uuid.UUID':>:
<class 'marshmallow.fields.UUID'>, <class 'datetime.time':>: <class
'marshmallow.fields.Time'>, <class 'datetime.date':>: <class
'marshmallow.fields.Date'>, <class 'datetime.timedelta':>: <class
'marshmallow.fields.TimeDelta'>, <class 'decimal.Decimal':>: <class
'marshmallow.fields.Decimal'>}
```

app.serializers.user.UserSerializer.dict_class

property UserSerializer.dict_class: type

app.serializers.user.UserSerializer.error_messages

UserSerializer.error_messages = {}
Overrides for default schema-level error messages

app.serializers.user.UserSerializer.opts

UserSerializer.opts = <marshmallow.schema.SchemaOpts object>

app.serializers.user.UserSerializer.set_class

property UserSerializer.set_class: type

Methods

<i>UserSerializer.__init__</i> (*[, only, exclude, ...])	
<i>UserSerializer.dump</i> (obj, *[, many])	Serialize an object to native Python data types according to this Schema's fields.
<i>UserSerializer.dumps</i> (obj, *args[, many])	Same as <i>dump()</i> , except return a JSON-encoded string.
<i>UserSerializer.from_dict</i> (fields, *[, name])	Generate a <i>Schema</i> class given a dictionary of fields.
<i>UserSerializer.get_attribute</i> (obj, attr, default)	Defines how to pull values from an object to serialize.
<i>UserSerializer.handle_error</i> (error, data, *, ...)	Custom error handler function for the schema.
<i>UserSerializer.jsonify</i> (obj[, many])	Return a JSON response containing the serialized data.
<i>UserSerializer.load</i> (data, *[, many, ...])	Deserialize a data structure to an object defined by this Schema's fields.
<i>UserSerializer.loads</i> (json_data, *[, many, ...])	Same as <i>load()</i> , except it takes a JSON string as input.
<i>UserSerializer.on_bind_field</i> (field_name, ...)	Hook to modify a field when it is bound to the <i>Schema</i> .
<i>UserSerializer.validate</i> (data, *[, many, partial])	Validate <i>data</i> against the schema, returning a dictionary of validation errors.
<i>UserSerializer.validate_email</i> (email)	
<i>UserSerializer.validate_id</i> (user_id)	

app.serializers.user.UserSerializer.__init__

`UserSerializer.__init__(*, only: Optional[Union[Sequence[str], Set[str]]] = None, exclude: Union[Sequence[str], Set[str]] = (), many: bool = False, context: Optional[Dict] = None, load_only: Union[Sequence[str], Set[str]] = (), dump_only: Union[Sequence[str], Set[str]] = (), partial: Union[bool, Sequence[str], Set[str]] = False, unknown: Optional[str] = None)`

app.serializers.user.UserSerializer.dump

`UserSerializer.dump(obj: Any, *, many: Optional[bool] = None)`

Serialize an object to native Python data types according to this Schema's fields.

Parameters

- **obj** – The object to serialize.
- **many** – Whether to serialize *obj* as a collection. If *None*, the value for *self.many* is used.

Returns Serialized data

New in version 1.0.0.

Changed in version 3.0.0b7: This method returns the serialized data rather than a (data, errors) tuple. A `ValidationError` is raised if *obj* is invalid.

Changed in version 3.0.0rc9: Validation no longer occurs upon serialization.

app.serializers.user.UserSerializer.dumps

`UserSerializer.dumps(obj: Any, *args, many: Optional[bool] = None, **kwargs)`

Same as `dump()`, except return a JSON-encoded string.

Parameters

- **obj** – The object to serialize.
- **many** – Whether to serialize *obj* as a collection. If *None*, the value for *self.many* is used.

Returns A json string

New in version 1.0.0.

Changed in version 3.0.0b7: This method returns the serialized data rather than a (data, errors) tuple. A `ValidationError` is raised if *obj* is invalid.

app.serializers.user.UserSerializer.from_dict

classmethod `UserSerializer.from_dict(fields: Dict[str, Union[marshmallow.fields.Field, type]], *, name: str = 'GeneratedSchema') → type`

Generate a *Schema* class given a dictionary of fields.

```
from marshmallow import Schema, fields

PersonSchema = Schema.from_dict({"name": fields.Str()})
print(PersonSchema().load({"name": "David"}))  # => {'name': 'David'}
```

Generated schemas are not added to the class registry and therefore cannot be referred to by name in *Nested* fields.

Parameters

- **fields** (*dict*) – Dictionary mapping field names to field instances.
- **name** (*str*) – Optional name for the class, which will appear in the `repr` for the class.

New in version 3.0.0.

app.serializers.user.UserSerializer.get_attribute

`UserSerializer.get_attribute(obj: Any, attr: str, default: Any)`

Defines how to pull values from an object to serialize.

New in version 2.0.0.

Changed in version 3.0.0a1: Changed position of `obj` and `attr`.

app.serializers.user.UserSerializer.handle_error

`UserSerializer.handle_error(error: marshmallow.exceptions.ValidationError, data: Any, *, many: bool, **kwargs)`

Custom error handler function for the schema.

Parameters

- **error** – The *ValidationError* raised during (de)serialization.
- **data** – The original input data.
- **many** – Value of `many` on dump or load.
- **partial** – Value of `partial` on load.

New in version 2.0.0.

Changed in version 3.0.0rc9: Receives *many* and *partial* (on deserialization) as keyword arguments.

app.serializers.user.UserSerializer.jsonify

`UserSerializer.jsonify(obj, many=<object object>, *args, **kwargs)`

Return a JSON response containing the serialized data.

Parameters

- **obj** – Object to serialize.
- **many** (*bool*) – Whether *obj* should be serialized as an instance or as a collection. If unset, defaults to the value of the *many* attribute on this Schema.
- **kwargs** – Additional keyword arguments passed to *flask.jsonify*.

Changed in version 0.6.0: Takes the same arguments as *marshmallow.Schema.dump*. Additional keyword arguments are passed to *flask.jsonify*.

Changed in version 0.6.3: The *many* argument for this method defaults to the value of the *many* attribute on the Schema. Previously, the *many* argument of this method defaulted to `False`, regardless of the value of *Schema.many*.

app.serializers.user.UserSerializer.load

`UserSerializer.load(data: Union[Mapping[str, Any], Iterable[Mapping[str, Any]]], *, many: Optional[bool] = None, partial: Optional[Union[bool, Sequence[str], Set[str]]] = None, unknown: Optional[str] = None)`

Deserialize a data structure to an object defined by this Schema's fields.

Parameters

- **data** – The data to deserialize.
- **many** – Whether to deserialize *data* as a collection. If *None*, the value for *self.many* is used.
- **partial** – Whether to ignore missing fields and not require any fields declared. Propagates down to *Nested* fields as well. If its value is an iterable, only missing fields listed in that iterable will be ignored. Use dot delimiters to specify nested fields.
- **unknown** – Whether to exclude, include, or raise an error for unknown fields in the data. Use *EXCLUDE*, *INCLUDE* or *RAISE*. If *None*, the value for *self.unknown* is used.

Returns Deserialized data

New in version 1.0.0.

Changed in version 3.0.0b7: This method returns the deserialized data rather than a (data, errors) tuple. A `ValidationError` is raised if invalid data are passed.

app.serializers.user.UserSerializer.loads

`UserSerializer.loads(json_data: str, *, many: Optional[bool] = None, partial: Optional[Union[bool, Sequence[str], Set[str]]] = None, unknown: Optional[str] = None, **kwargs)`

Same as `load()`, except it takes a JSON string as input.

Parameters

- **json_data** – A JSON string of the data to deserialize.
- **many** – Whether to deserialize *obj* as a collection. If *None*, the value for *self.many* is used.
- **partial** – Whether to ignore missing fields and not require any fields declared. Propagates down to *Nested* fields as well. If its value is an iterable, only missing fields listed in that iterable will be ignored. Use dot delimiters to specify nested fields.
- **unknown** – Whether to exclude, include, or raise an error for unknown fields in the data. Use *EXCLUDE*, *INCLUDE* or *RAISE*. If *None*, the value for *self.unknown* is used.

Returns Deserialized data

New in version 1.0.0.

Changed in version 3.0.0b7: This method returns the deserialized data rather than a (data, errors) tuple. A `ValidationError` is raised if invalid data are passed.

app.serializers.user.UserSerializer.on_bind_field

`UserSerializer.on_bind_field(field_name: str, field_obj: marshmallow.fields.Field) → None`

Hook to modify a field when it is bound to the *Schema*.

No-op by default.

app.serializers.user.UserSerializer.validate

`UserSerializer.validate(data: Union[Mapping[str, Any], Iterable[Mapping[str, Any]]], *, many: Optional[bool] = None, partial: Optional[Union[bool, Sequence[str], Set[str]]] = None) → Dict[str, List[str]]`

Validate *data* against the schema, returning a dictionary of validation errors.

Parameters

- **data** – The data to validate.
- **many** – Whether to validate *data* as a collection. If *None*, the value for *self.many* is used.
- **partial** – Whether to ignore missing fields and not require any fields declared. Propagates down to *Nested* fields as well. If its value is an iterable, only missing fields listed in that iterable will be ignored. Use dot delimiters to specify nested fields.

Returns A dictionary of validation errors.

New in version 1.1.0.

app.serializers.user.UserSerializer.validate_email

`UserSerializer.validate_email(email: str)`

app.serializers.user.UserSerializer.validate_id

`UserSerializer.validate_id(user_id: int)`

app.serializers.user.VerifyRoleId

```
class app.serializers.user.VerifyRoleId(*, load_default: Any = <marshmallow.missing>, missing: Any =
    <marshmallow.missing>, dump_default: Any =
    <marshmallow.missing>, default: Any =
    <marshmallow.missing>, data_key: Optional[str] = None,
    attribute: Optional[str] = None, validate:
    Optional[Union[Callable[[Any], Any], Iterable[Callable[[Any],
    Any]]]] = None, required: bool = False, allow_none:
    Optional[bool] = None, load_only: bool = False, dump_only:
    bool = False, error_messages: Optional[Dict[str, str]] = None,
    metadata: Optional[Mapping[str, Any]] = None,
    **additional_metadata)
```

Bases: `marshmallow.fields.Field`

Attributes

<i>VerifyRoleId.context</i>	The context dictionary for the parent Schema.
<i>VerifyRoleId.default</i>	
<i>VerifyRoleId.default_error_messages</i>	Default error messages for various kinds of errors.
<i>VerifyRoleId.missing</i>	
<i>VerifyRoleId.name</i>	
<i>VerifyRoleId.parent</i>	
<i>VerifyRoleId.root</i>	

app.serializers.user.VerifyRoleId.context

property `VerifyRoleId.context`
The context dictionary for the parent Schema.

app.serializers.user.VerifyRoleId.default

property `VerifyRoleId.default`

app.serializers.user.VerifyRoleId.default_error_messages

`VerifyRoleId.default_error_messages = {'null': 'Field may not be null.', 'required': 'Missing data for required field.', 'validator_failed': 'Invalid value.'}`
Default error messages for various kinds of errors. The keys in this dictionary are passed to *Field.make_error*. The values are error messages passed to *marshmallow.exceptions.ValidationError*.

app.serializers.user.VerifyRoleId.missing

property `VerifyRoleId.missing`

app.serializers.user.VerifyRoleId.name

`VerifyRoleId.name = None`

app.serializers.user.VerifyRoleId.parent

`VerifyRoleId.parent = None`

app.serializers.user.VerifyRoleId.root

`VerifyRoleId.root = None`

Methods

<code>VerifyRoleId.__init__(*[, load_default, ...])</code>	
<code>VerifyRoleId.deserialize(value[, attr, data])</code>	Deserialize value.
<code>VerifyRoleId.fail(key, **kwargs)</code>	Helper method that raises a <i>ValidationError</i> with an error message from <code>self.error_messages</code> .
<code>VerifyRoleId.get_value(obj, attr[, ...])</code>	Return the value for a given key from an object.
<code>VerifyRoleId.make_error(key, **kwargs)</code>	Helper method to make a <i>ValidationError</i> with an error message from <code>self.error_messages</code> .
<code>VerifyRoleId.serialize(attr, obj[, accessor])</code>	Pulls the value for the given key from the object, applies the field's formatting and returns the result.

app.serializers.user.VerifyRoleId.__init__

`VerifyRoleId.__init__(*, load_default: Any = <marshmallow.missing>, missing: Any = <marshmallow.missing>, dump_default: Any = <marshmallow.missing>, default: Any = <marshmallow.missing>, data_key: Optional[str] = None, attribute: Optional[str] = None, validate: Optional[Union[Callable[[Any], Any], Iterable[Callable[[Any], Any]]]] = None, required: bool = False, allow_none: Optional[bool] = None, load_only: bool = False, dump_only: bool = False, error_messages: Optional[Dict[str, str]] = None, metadata: Optional[Mapping[str, Any]] = None, **additional_metadata) → None`

app.serializers.user.VerifyRoleId.deserialize

`VerifyRoleId.deserialize(value: Any, attr: Optional[str] = None, data: Optional[Mapping[str, Any]] = None, **kwargs)`

Deserialize value.

Parameters

- **value** – The value to deserialize.
- **attr** – The attribute/key in `data` to deserialize.
- **data** – The raw input data passed to `Schema.load`.
- **kwargs** – Field-specific keyword arguments.

Raises `ValidationError` – If an invalid value is passed or if a required value is missing.

`app.serializers.user.VerifyRoleId.fail`

`VerifyRoleId.fail(key: str, **kwargs)`

Helper method that raises a *ValidationError* with an error message from `self.error_messages`.

Deprecated since version 3.0.0: Use *make_error* <marshmallow.fields.Field.make_error> instead.

`app.serializers.user.VerifyRoleId.get_value`

`VerifyRoleId.get_value(obj, attr, accessor=None, default=<marshmallow.missing>)`

Return the value for a given key from an object.

Parameters

- **obj** (*object*) – The object to get the value from.
- **attr** (*str*) – The attribute/key in *obj* to get the value from.
- **accessor** (*callable*) – A callable used to retrieve the value of *attr* from the object *obj*. Defaults to *marshmallow.utils.get_value*.

`app.serializers.user.VerifyRoleId.make_error`

`VerifyRoleId.make_error(key: str, **kwargs) → marshmallow.exceptions.ValidationError`

Helper method to make a *ValidationError* with an error message from `self.error_messages`.

`app.serializers.user.VerifyRoleId.serialize`

`VerifyRoleId.serialize(attr: str, obj: Any, accessor: Optional[Callable[[Any, str, Any], Any]] = None, **kwargs)`

Pulls the value for the given key from the object, applies the field's formatting and returns the result.

Parameters

- **attr** – The attribute/key to get from the object.
- **obj** – The object to access the attribute/key from.
- **accessor** – Function used to access values from *obj*.
- **kwargs** – Field-specific keyword arguments.

```
class app.serializers.user.UserExportWordSerializer(*, only: Optional[Union[Sequence[str],
Set[str]]] = None, exclude:
    Union[Sequence[str], Set[str]] = (), many: bool
    = False, context: Optional[Dict] = None,
    load_only: Union[Sequence[str], Set[str]] = (),
    dump_only: Union[Sequence[str], Set[str]] = (),
    partial: Union[bool, Sequence[str], Set[str]] =
    False, unknown: Optional[str] = None)
```

`class Meta`

Options object for a Schema.

Example usage:

```
class Meta:
    fields = ("id", "email", "date_created")
    exclude = ("password", "secret_attribute")
```

Available options:

- **fields:** Tuple or list of fields to include in the serialized result.
- **additional:** Tuple or list of fields to include *in addition to the* explicitly declared fields. **additional** and **fields** are mutually-exclusive options.
- **include:** Dictionary of additional fields to include in the schema. **It is** usually better to define fields as class variables, but you may need to use this option, e.g., if your fields are Python keywords. May be an *OrderedDict*.
- **exclude:** Tuple or list of fields to exclude in the serialized result. Nested fields can be represented with dot delimiters.
- **dateformat:** Default format for *Date* <fields.Date> fields.
- **datetimeformat:** Default format for *DateTime* <fields.DateTime> fields.
- **timeformat:** Default format for *Time* <fields.Time> fields.
- **render_module:** Module to use for *loads* <Schema.loads> and *dumps* <Schema.dumps>. Defaults to *json* from the standard library.
- **ordered:** If *True*, order serialization output according to the order in which fields were declared. Output of *Schema.dump* will be a *collections.OrderedDict*.
- **index_errors:** If *True*, errors dictionaries will include the **index** of invalid items in a collection.
- **load_only:** Tuple or list of fields to exclude from serialized results.
- **dump_only:** Tuple or list of fields to exclude from deserialization
- **unknown:** Whether to exclude, include, or raise an error for unknown fields in the data. Use *EXCLUDE*, *INCLUDE* or *RAISE*.
- **register:** Whether to register the *Schema* with *marshmallow's* internal class registry. Must be *True* if you intend to refer to this *Schema* by class name in *Nested* fields. Only set this to *False* when memory usage is critical. Defaults to *True*.

OPTIONS_CLASS

alias of `marshmallow.schema.SchemaOpts`

```
TYPE_MAPPING = {<class 'str':>: <class 'marshmallow.fields.String'>, <class
'bytes':>: <class 'marshmallow.fields.String'>, <class 'datetime.datetime':>: <class
'marshmallow.fields.DateTime'>, <class 'float':>: <class
'marshmallow.fields.Float'>, <class 'bool':>: <class 'marshmallow.fields.Boolean'>,
<class 'tuple':>: <class 'marshmallow.fields.Raw'>, <class 'list':>: <class
'marshmallow.fields.Raw'>, <class 'set':>: <class 'marshmallow.fields.Raw'>, <class
'int':>: <class 'marshmallow.fields.Integer'>, <class 'uuid.UUID':>: <class
'marshmallow.fields.UUID'>, <class 'datetime.time':>: <class
'marshmallow.fields.Time'>, <class 'datetime.date':>: <class
'marshmallow.fields.Date'>, <class 'datetime.timedelta':>: <class
'marshmallow.fields.TimeDelta'>, <class 'decimal.Decimal':>: <class
'marshmallow.fields.Decimal'>}
```

_bind_field(*field_name*: str, *field_obj*: *marshmallow.fields.Field*) → None

Bind field to the schema, setting any necessary attributes on the field (e.g. parent and name).

Also set field *load_only* and *dump_only* values if *field_name* was specified in class *Meta*.

static _call_and_store(*getter_func*, *data*, *, *field_name*, *error_store*, *index*=None)

Call *getter_func* with *data* as its argument, and store any *ValidationErrors*.

Parameters

- **getter_func** (*callable*) – Function for getting the serialized/deserialized value from *data*.
- **data** – The data passed to *getter_func*.
- **field_name** (*str*) – Field name.
- **index** (*int*) – Index of the item being validated, if validating a collection, otherwise *None*.

```
_declared_fields = {'to_pdf': <fields.Integer(dump_default=<marshmallow.missing>,
attribute=None, validate=<OneOf(choices=[1, 0], labels=[], error='Must be one of:
{choices}.')>, required=False, load_only=False, dump_only=False,
load_default=<marshmallow.missing>, allow_none=False, error_messages={'required':
'Missing data for required field.', 'null': 'Field may not be null.',
'validator_failed': 'Invalid value.', 'invalid': 'Not a valid integer.',
'too_large': 'Number too large.'})>>}
```

```
_default_error_messages = {'type': 'Invalid input type.', 'unknown': 'Unknown
field.'}
```

```
_deserialize(data: Union[Mapping[str, Any], Iterable[Mapping[str, Any]]], *, error_store:
marshmallow.error_store.ErrorStore, many: bool = False, partial=False, unknown='raise',
index=None) → Union[marshmallow.schema._T, List[marshmallow.schema._T]]
```

Deserialize data.

Parameters

- **data** (*dict*) – The data to deserialize.
- **error_store** (*ErrorStore*) – Structure to store errors.
- **many** (*bool*) – *True* if data should be deserialized as a collection.
- **partial** (*bool* | *tuple*) – Whether to ignore missing fields and not require any fields declared. Propagates down to Nested fields as well. If its value is an iterable, only missing fields listed in that iterable will be ignored. Use dot delimiters to specify nested fields.
- **unknown** – Whether to exclude, include, or raise an error for unknown fields in the data. Use *EXCLUDE*, *INCLUDE* or *RAISE*.
- **index** (*int*) – Index of the item being serialized (for storing errors) if serializing a collection, otherwise *None*.

Returns A dictionary of the deserialized data.

```
_do_load(data: Union[Mapping[str, Any], Iterable[Mapping[str, Any]]], *, many: Optional[bool] = None,
partial: Optional[Union[bool, Sequence[str], Set[str]]] = None, unknown: Optional[str] = None,
postprocess: bool = True)
```

Deserialize *data*, returning the deserialized result. This method is private API.

Parameters

- **data** – The data to deserialize.
- **many** – Whether to deserialize *data* as a collection. If *None*, the value for *self.many* is used.

- **partial** – Whether to validate required fields. If its value is an iterable, only fields listed in that iterable will be ignored will be allowed missing. If *True*, all fields will be allowed missing. If *None*, the value for *self.partial* is used.
- **unknown** – Whether to exclude, include, or raise an error for unknown fields in the data. Use *EXCLUDE*, *INCLUDE* or *RAISE*. If *None*, the value for *self.unknown* is used.
- **postprocess** – Whether to run *post_load* methods..

Returns Deserialized data

_has_processors(*tag*) → bool

_hooks = {'pre_load', False): ['process_input']}

_init_fields() → None

Update *self.fields*, *self.load_fields*, and *self.dump_fields* based on schema options. This method is private API.

_invoke_dump_processors(*tag: str*, *data*, *, *many: bool*, *original_data=None*)

_invoke_field_validators(*, *error_store: marshmallow.error_store.ErrorStore*, *data*, *many: bool*)

_invoke_load_processors(*tag: str*, *data*, *, *many: bool*, *original_data*, *partial: Union[bool, Sequence[str], Set[str]]*)

_invoke_processors(*tag: str*, *, *pass_many: bool*, *data*, *many: bool*, *original_data=None*, **kwargs)

_invoke_schema_validators(*, *error_store: marshmallow.error_store.ErrorStore*, *pass_many: bool*, *data*, *original_data*, *many: bool*, *partial: Union[bool, Sequence[str], Set[str]]*, *field_errors: bool = False*)

_normalize_nested_options() → None

Apply then flatten nested schema options. This method is private API.

_run_validator(*validator_func*, *output*, *, *original_data*, *error_store*, *many*, *partial*, *pass_original*, *index=None*)

_serialize(*obj: Union[marshmallow.schema._T, Iterable[marshmallow.schema._T]]*, *, *many: bool = False*)

Serialize *obj*.

Parameters

- **obj** – The object(s) to serialize.
- **many** (*bool*) – *True* if data should be serialized as a collection.

Returns A dictionary of the serialized data

Changed in version 1.0.0: Renamed from *marshal*.

property dict_class: type

dump(*obj: Any*, *, *many: Optional[bool] = None*)

Serialize an object to native Python data types according to this Schema's fields.

Parameters

- **obj** – The object to serialize.
- **many** – Whether to serialize *obj* as a collection. If *None*, the value for *self.many* is used.

Returns Serialized data

New in version 1.0.0.

Changed in version 3.0.0b7: This method returns the serialized data rather than a (data, errors) tuple. A `ValidationError` is raised if `obj` is invalid.

Changed in version 3.0.0rc9: Validation no longer occurs upon serialization.

dumps(*obj: Any, *args, many: Optional[bool] = None, **kwargs*)

Same as `dump()`, except return a JSON-encoded string.

Parameters

- **obj** – The object to serialize.
- **many** – Whether to serialize *obj* as a collection. If *None*, the value for *self.many* is used.

Returns A json string

New in version 1.0.0.

Changed in version 3.0.0b7: This method returns the serialized data rather than a (data, errors) tuple. A `ValidationError` is raised if `obj` is invalid.

error_messages = {}

Overrides for default schema-level error messages

fields

Dictionary mapping field_names -> Field objects

classmethod from_dict(*fields: Dict[str, Union[marshmallow.fields.Field, type]], *, name: str = 'GeneratedSchema') → type*

Generate a *Schema* class given a dictionary of fields.

```
from marshmallow import Schema, fields

PersonSchema = Schema.from_dict({"name": fields.Str()})
print(PersonSchema().load({"name": "David"})) # => {'name': 'David'}
```

Generated schemas are not added to the class registry and therefore cannot be referred to by name in *Nested* fields.

Parameters

- **fields** (*dict*) – Dictionary mapping field names to field instances.
- **name** (*str*) – Optional name for the class, which will appear in the `repr` for the class.

New in version 3.0.0.

get_attribute(*obj: Any, attr: str, default: Any*)

Defines how to pull values from an object to serialize.

New in version 2.0.0.

Changed in version 3.0.0a1: Changed position of `obj` and `attr`.

handle_error(*error: marshmallow.exceptions.ValidationError, data: Any, *, many: bool, **kwargs*)

Custom error handler function for the schema.

Parameters

- **error** – The *ValidationError* raised during (de)serialization.
- **data** – The original input data.
- **many** – Value of `many` on `dump` or `load`.

- **partial** – Value of **partial** on load.

New in version 2.0.0.

Changed in version 3.0.0rc9: Receives *many* and *partial* (on deserialization) as keyword arguments.

jsonify(*obj*, *many*=<object object>, **args*, ***kwargs*)

Return a JSON response containing the serialized data.

Parameters

- **obj** – Object to serialize.
- **many** (*bool*) – Whether *obj* should be serialized as an instance or as a collection. If unset, defaults to the value of the *many* attribute on this Schema.
- **kwargs** – Additional keyword arguments passed to *flask jsonify*.

Changed in version 0.6.0: Takes the same arguments as *marshmallow.Schema.dump*. Additional keyword arguments are passed to *flask jsonify*.

Changed in version 0.6.3: The *many* argument for this method defaults to the value of the *many* attribute on the Schema. Previously, the *many* argument of this method defaulted to False, regardless of the value of *Schema.many*.

load(*data*: Union[Mapping[str, Any], Iterable[Mapping[str, Any]]], *, *many*: Optional[bool] = None, *partial*: Optional[Union[bool, Sequence[str], Set[str]]] = None, *unknown*: Optional[str] = None)

Deserialize a data structure to an object defined by this Schema's fields.

Parameters

- **data** – The data to deserialize.
- **many** – Whether to deserialize *data* as a collection. If *None*, the value for *self.many* is used.
- **partial** – Whether to ignore missing fields and not require any fields declared. Propagates down to *Nested* fields as well. If its value is an iterable, only missing fields listed in that iterable will be ignored. Use dot delimiters to specify nested fields.
- **unknown** – Whether to exclude, include, or raise an error for unknown fields in the data. Use *EXCLUDE*, *INCLUDE* or *RAISE*. If *None*, the value for *self.unknown* is used.

Returns Deserialized data

New in version 1.0.0.

Changed in version 3.0.0b7: This method returns the deserialized data rather than a (data, errors) tuple. A *ValidationError* is raised if invalid data are passed.

loads(*json_data*: str, *, *many*: Optional[bool] = None, *partial*: Optional[Union[bool, Sequence[str], Set[str]]] = None, *unknown*: Optional[str] = None, ***kwargs*)

Same as *load()*, except it takes a JSON string as input.

Parameters

- **json_data** – A JSON string of the data to deserialize.
- **many** – Whether to deserialize *obj* as a collection. If *None*, the value for *self.many* is used.
- **partial** – Whether to ignore missing fields and not require any fields declared. Propagates down to *Nested* fields as well. If its value is an iterable, only missing fields listed in that iterable will be ignored. Use dot delimiters to specify nested fields.
- **unknown** – Whether to exclude, include, or raise an error for unknown fields in the data. Use *EXCLUDE*, *INCLUDE* or *RAISE*. If *None*, the value for *self.unknown* is used.

Returns Deserialized data

New in version 1.0.0.

Changed in version 3.0.0b7: This method returns the deserialized data rather than a (data, errors) tuple. A `ValidationError` is raised if invalid data are passed.

on_bind_field(*field_name: str, field_obj: marshmallow.fields.Field*) → None

Hook to modify a field when it is bound to the *Schema*.

No-op by default.

opts = <marshmallow.schema.SchemaOpts object>

process_input(*value, many, **kwargs*)

property set_class: type

validate(*data: Union[Mapping[str, Any], Iterable[Mapping[str, Any]]], *, many: Optional[bool] = None, partial: Optional[Union[bool, Sequence[str], Set[str]]] = None*) → Dict[str, List[str]]

Validate *data* against the schema, returning a dictionary of validation errors.

Parameters

- **data** – The data to validate.
- **many** – Whether to validate *data* as a collection. If *None*, the value for *self.many* is used.
- **partial** – Whether to ignore missing fields and not require any fields declared. Propagates down to *Nested* fields as well. If its value is an iterable, only missing fields listed in that iterable will be ignored. Use dot delimiters to specify nested fields.

Returns A dictionary of validation errors.

New in version 1.1.0.

```
class app.serializers.user.UserSerializer(*, only: Optional[Union[Sequence[str], Set[str]]] = None,
                                          exclude: Union[Sequence[str], Set[str]] = (), many: bool =
                                          False, context: Optional[Dict] = None, load_only:
                                          Union[Sequence[str], Set[str]] = (), dump_only:
                                          Union[Sequence[str], Set[str]] = (), partial: Union[bool,
                                          Sequence[str], Set[str]] = False, unknown: Optional[str] =
                                          None)
```

class Meta

ordered = True

OPTIONS_CLASS

alias of marshmallow.schema.SchemaOpts

```
TYPE_MAPPING = {<class 'str':>: <class 'marshmallow.fields.String'>, <class  
'bytes':>: <class 'marshmallow.fields.String'>, <class 'datetime.datetime':>: <class  
'marshmallow.fields.DateTime'>, <class 'float':>: <class  
'marshmallow.fields.Float'>, <class 'bool':>: <class 'marshmallow.fields.Boolean'>,  
<class 'tuple':>: <class 'marshmallow.fields.Raw'>, <class 'list':>: <class  
'marshmallow.fields.Raw'>, <class 'set':>: <class 'marshmallow.fields.Raw'>, <class  
'int':>: <class 'marshmallow.fields.Integer'>, <class 'uuid.UUID':>: <class  
'marshmallow.fields.UUID'>, <class 'datetime.time':>: <class  
'marshmallow.fields.Time'>, <class 'datetime.date':>: <class  
'marshmallow.fields.Date'>, <class 'datetime.timedelta':>: <class  
'marshmallow.fields.TimeDelta'>, <class 'decimal.Decimal':>: <class  
'marshmallow.fields.Decimal'>}
```

`_bind_field`(*field_name*: str, *field_obj*: marshmallow.fields.Field) → None

Bind field to the schema, setting any necessary attributes on the field (e.g. parent and name).

Also set field `load_only` and `dump_only` values if `field_name` was specified in class `Meta`.

`static _call_and_store`(*getter_func*, *data*, *, *field_name*, *error_store*, *index*=None)

Call `getter_func` with `data` as its argument, and store any `ValidationErrors`.

Parameters

- **`getter_func`** (*callable*) – Function for getting the serialized/deserialized value from data.
- **`data`** – The data passed to `getter_func`.
- **`field_name`** (*str*) – Field name.
- **`index`** (*int*) – Index of the item being validated, if validating a collection, otherwise *None*.

```

_declared_fields = {'active': <fields.Boolean(dump_default=<marshmallow.missing>,
attribute=None, validate=None, required=False, load_only=False, dump_only=False,
load_default=<marshmallow.missing>, allow_none=False, error_messages={'required':
'Missing data for required field.', 'null': 'Field may not be null.',
'validator_failed': 'Invalid value.', 'invalid': 'Not a valid boolean.'})>,
'birth_date': <fields.Date(dump_default=<marshmallow.missing>, attribute=None,
validate=None, required=False, load_only=False, dump_only=False,
load_default=<marshmallow.missing>, allow_none=False, error_messages={'required':
'Missing data for required field.', 'null': 'Field may not be null.',
'validator_failed': 'Invalid value.', 'invalid': 'Not a valid date.',
'invalid_awareness': 'Not a valid {awareness} {obj_type}.'. , 'format': '"{input}"
cannot be formatted as a date.'})>, 'created_at':
<fields.TimestampField(dump_default=<marshmallow.missing>, attribute=None,
validate=None, required=False, load_only=False, dump_only=True,
load_default=<marshmallow.missing>, allow_none=False, error_messages={'required':
'Missing data for required field.', 'null': 'Field may not be null.',
'validator_failed': 'Invalid value.'})>, 'created_by':
<fields.Nested(dump_default=<marshmallow.missing>, attribute=None, validate=None,
required=False, load_only=False, dump_only=False,
load_default=<marshmallow.missing>, allow_none=False, error_messages={'required':
'Missing data for required field.', 'null': 'Field may not be null.',
'validator_failed': 'Invalid value.', 'type': 'Invalid type.'})>, 'deleted_at':
<fields.TimestampField(dump_default=<marshmallow.missing>, attribute=None,
validate=None, required=False, load_only=False, dump_only=True,
load_default=<marshmallow.missing>, allow_none=False, error_messages={'required':
'Missing data for required field.', 'null': 'Field may not be null.',
'validator_failed': 'Invalid value.'})>, 'email':
<fields.Email(dump_default=<marshmallow.missing>, attribute=None, validate=None,
required=True, load_only=False, dump_only=False, load_default=<marshmallow.missing>,
allow_none=False, error_messages={'required': 'Missing data for required field.',
'null': 'Field may not be null.', 'validator_failed': 'Invalid value.', 'invalid':
'Not a valid email address.', 'invalid_utf8': 'Not a valid utf-8 string.'})>,
'genre': <fields.String(dump_default=<marshmallow.missing>, attribute=None,
validate=<OneOf(choices=['m', 'f'], labels=[], error='Must be one of:
{choices}.')>, required=False, load_only=False, dump_only=False,
load_default=<marshmallow.missing>, allow_none=False, error_messages={'required':
'Missing data for required field.', 'null': 'Field may not be null.',
'validator_failed': 'Invalid value.', 'invalid': 'Not a valid string.',
'invalid_utf8': 'Not a valid utf-8 string.'})>, 'id':
<fields.Integer(dump_default=<marshmallow.missing>, attribute=None, validate=None,
required=False, load_only=False, dump_only=False,
load_default=<marshmallow.missing>, allow_none=False, error_messages={'required':
'Missing data for required field.', 'null': 'Field may not be null.',
'validator_failed': 'Invalid value.', 'invalid': 'Not a valid integer.',
'too_large': 'Number too large.'})>, 'last_name':
<fields.String(dump_default=<marshmallow.missing>, attribute=None, validate=None,
required=False, load_only=False, dump_only=False,
load_default=<marshmallow.missing>, allow_none=False, error_messages={'required':
'Missing data for required field.', 'null': 'Field may not be null.',
'validator_failed': 'Invalid value.', 'invalid': 'Not a valid string.',
'invalid_utf8': 'Not a valid utf-8 string.'})>, 'name':
<fields.String(dump_default=<marshmallow.missing>, attribute=None, validate=None,
required=False, load_only=False, dump_only=False,
load_default=<marshmallow.missing>, allow_none=False, error_messages={'required':
'Missing data for required field.', 'null': 'Field may not be null.',
'validator_failed': 'Invalid value.', 'invalid': 'Not a valid string.',
'invalid_utf8': 'Not a valid utf-8 string.'})>, 'password':
<fields.String(dump_default=<marshmallow.missing>, attribute=None,
validate=<Length(min=8, max=50, equal=None, error=None)>, required=False,
load_only=True, dump_only=False, load_default=<marshmallow.missing>,

```

```
_default_error_messages = {'type': 'Invalid input type.', 'unknown': 'Unknown field.'}
```

```
_deserialize(data: Union[Mapping[str, Any], Iterable[Mapping[str, Any]]], *, error_store:
    marshmallow.error_store.ErrorStore, many: bool = False, partial=False, unknown='raise',
    index=None) → Union[marshmallow.schema._T, List[marshmallow.schema._T]]
```

Deserialize data.

Parameters

- **data** (*dict*) – The data to deserialize.
- **error_store** (*ErrorStore*) – Structure to store errors.
- **many** (*bool*) – *True* if data should be deserialized as a collection.
- **partial** (*bool | tuple*) – Whether to ignore missing fields and not require any fields declared. Propagates down to *Nested* fields as well. If its value is an iterable, only missing fields listed in that iterable will be ignored. Use dot delimiters to specify nested fields.
- **unknown** – Whether to exclude, include, or raise an error for unknown fields in the data. Use *EXCLUDE*, *INCLUDE* or *RAISE*.
- **index** (*int*) – Index of the item being serialized (for storing errors) if serializing a collection, otherwise *None*.

Returns A dictionary of the deserialized data.

```
_do_load(data: Union[Mapping[str, Any], Iterable[Mapping[str, Any]]], *, many: Optional[bool] = None,
    partial: Optional[Union[bool, Sequence[str], Set[str]]] = None, unknown: Optional[str] = None,
    postprocess: bool = True)
```

Deserialize *data*, returning the deserialized result. This method is private API.

Parameters

- **data** – The data to deserialize.
- **many** – Whether to deserialize *data* as a collection. If *None*, the value for *self.many* is used.
- **partial** – Whether to validate required fields. If its value is an iterable, only fields listed in that iterable will be ignored will be allowed missing. If *True*, all fields will be allowed missing. If *None*, the value for *self.partial* is used.
- **unknown** – Whether to exclude, include, or raise an error for unknown fields in the data. Use *EXCLUDE*, *INCLUDE* or *RAISE*. If *None*, the value for *self.unknown* is used.
- **postprocess** – Whether to run *post_load* methods..

Returns Deserialized data

```
_has_processors(tag) → bool
```

```
_hooks = {'validates': ['validate_email', 'validate_id']}
```

```
_init_fields() → None
```

Update *self.fields*, *self.load_fields*, and *self.dump_fields* based on schema options. This method is private API.

```
_invoke_dump_processors(tag: str, data, *, many: bool, original_data=None)
```

```
_invoke_field_validators(*, error_store: marshmallow.error_store.ErrorStore, data, many: bool)
```

```
_invoke_load_processors(tag: str, data, *, many: bool, original_data, partial: Union[bool,
    Sequence[str], Set[str]])
```

```
_invoke_processors(tag: str, *, pass_many: bool, data, many: bool, original_data=None, **kwargs)
```

_invoke_schema_validators(**error_store: marshmallow.error_store.ErrorStore*, *pass_many: bool*, *data*, *original_data*, *many: bool*, *partial: Union[bool, Sequence[str], Set[str]]*, *field_errors: bool = False*)

_normalize_nested_options() → None

Apply then flatten nested schema options. This method is private API.

_run_validator(*validator_func*, *output*, *, *original_data*, *error_store*, *many*, *partial*, *pass_original*, *index=None*)

_serialize(*obj: Union[marshmallow.schema._T, Iterable[marshmallow.schema._T]]*, *, *many: bool = False*)

Serialize obj.

Parameters

- **obj** – The object(s) to serialize.
- **many** (*bool*) – *True* if data should be serialized as a collection.

Returns A dictionary of the serialized data

Changed in version 1.0.0: Renamed from `marshal`.

property dict_class: type

dump(*obj: Any*, *, *many: Optional[bool] = None*)

Serialize an object to native Python data types according to this Schema's fields.

Parameters

- **obj** – The object to serialize.
- **many** – Whether to serialize *obj* as a collection. If *None*, the value for *self.many* is used.

Returns Serialized data

New in version 1.0.0.

Changed in version 3.0.0b7: This method returns the serialized data rather than a (*data*, *errors*) tuple. A `ValidationError` is raised if *obj* is invalid.

Changed in version 3.0.0rc9: Validation no longer occurs upon serialization.

dumps(*obj: Any*, **args*, *many: Optional[bool] = None*, ***kwargs*)

Same as `dump()`, except return a JSON-encoded string.

Parameters

- **obj** – The object to serialize.
- **many** – Whether to serialize *obj* as a collection. If *None*, the value for *self.many* is used.

Returns A json string

New in version 1.0.0.

Changed in version 3.0.0b7: This method returns the serialized data rather than a (*data*, *errors*) tuple. A `ValidationError` is raised if *obj* is invalid.

error_messages = {}

Overrides for default schema-level error messages

fields

Dictionary mapping field_names -> Field objects

classmethod from_dict(fields: Dict[str, Union[marshmallow.fields.Field, type]], *, name: str = 'GeneratedSchema') → type
Generate a *Schema* class given a dictionary of fields.

```
from marshmallow import Schema, fields

PersonSchema = Schema.from_dict({"name": fields.Str()})
print(PersonSchema().load({"name": "David"})) # => {'name': 'David'}
```

Generated schemas are not added to the class registry and therefore cannot be referred to by name in *Nested* fields.

Parameters

- **fields** (*dict*) – Dictionary mapping field names to field instances.
- **name** (*str*) – Optional name for the class, which will appear in the `repr` for the class.

New in version 3.0.0.

get_attribute(obj: Any, attr: str, default: Any)
Defines how to pull values from an object to serialize.

New in version 2.0.0.

Changed in version 3.0.0a1: Changed position of `obj` and `attr`.

handle_error(error: marshmallow.exceptions.ValidationError, data: Any, *, many: bool, **kwargs)
Custom error handler function for the schema.

Parameters

- **error** – The *ValidationError* raised during (de)serialization.
- **data** – The original input data.
- **many** – Value of `many` on dump or load.
- **partial** – Value of `partial` on load.

New in version 2.0.0.

Changed in version 3.0.0rc9: Receives *many* and *partial* (on deserialization) as keyword arguments.

jsonify(obj, many=<object object>, *args, **kwargs)
Return a JSON response containing the serialized data.

Parameters

- **obj** – Object to serialize.
- **many** (*bool*) – Whether *obj* should be serialized as an instance or as a collection. If unset, defaults to the value of the *many* attribute on this Schema.
- **kwargs** – Additional keyword arguments passed to *flask.jsonify*.

Changed in version 0.6.0: Takes the same arguments as *marshmallow.Schema.dump*. Additional keyword arguments are passed to *flask.jsonify*.

Changed in version 0.6.3: The *many* argument for this method defaults to the value of the *many* attribute on the Schema. Previously, the *many* argument of this method defaulted to `False`, regardless of the value of *Schema.many*.

load(data: Union[Mapping[str, Any], Iterable[Mapping[str, Any]]], *, many: Optional[bool] = None, partial: Optional[Union[bool, Sequence[str], Set[str]]] = None, unknown: Optional[str] = None)
Deserialize a data structure to an object defined by this Schema's fields.

Parameters

- **data** – The data to deserialize.
- **many** – Whether to deserialize *data* as a collection. If *None*, the value for *self.many* is used.
- **partial** – Whether to ignore missing fields and not require any fields declared. Propagates down to *Nested* fields as well. If its value is an iterable, only missing fields listed in that iterable will be ignored. Use dot delimiters to specify nested fields.
- **unknown** – Whether to exclude, include, or raise an error for unknown fields in the data. Use *EXCLUDE*, *INCLUDE* or *RAISE*. If *None*, the value for *self.unknown* is used.

Returns Deserialized data

New in version 1.0.0.

Changed in version 3.0.0b7: This method returns the deserialized data rather than a (data, errors) tuple. A *ValidationError* is raised if invalid data are passed.

loads(*json_data: str*, *, *many: Optional[bool] = None*, *partial: Optional[Union[bool, Sequence[str], Set[str]]] = None*, *unknown: Optional[str] = None*, ***kwargs*)
Same as *load()*, except it takes a JSON string as input.

Parameters

- **json_data** – A JSON string of the data to deserialize.
- **many** – Whether to deserialize *obj* as a collection. If *None*, the value for *self.many* is used.
- **partial** – Whether to ignore missing fields and not require any fields declared. Propagates down to *Nested* fields as well. If its value is an iterable, only missing fields listed in that iterable will be ignored. Use dot delimiters to specify nested fields.
- **unknown** – Whether to exclude, include, or raise an error for unknown fields in the data. Use *EXCLUDE*, *INCLUDE* or *RAISE*. If *None*, the value for *self.unknown* is used.

Returns Deserialized data

New in version 1.0.0.

Changed in version 3.0.0b7: This method returns the deserialized data rather than a (data, errors) tuple. A *ValidationError* is raised if invalid data are passed.

on_bind_field(*field_name: str*, *field_obj: marshmallow.fields.Field*) → *None*
Hook to modify a field when it is bound to the *Schema*.

No-op by default.

opts = <marshmallow.schema.SchemaOpts object>

property set_class: type

validate(*data: Union[Mapping[str, Any], Iterable[Mapping[str, Any]]]*, *, *many: Optional[bool] = None*, *partial: Optional[Union[bool, Sequence[str], Set[str]]] = None*) → *Dict[str, List[str]]*
Validate *data* against the schema, returning a dictionary of validation errors.

Parameters

- **data** – The data to validate.
- **many** – Whether to validate *data* as a collection. If *None*, the value for *self.many* is used.
- **partial** – Whether to ignore missing fields and not require any fields declared. Propagates down to *Nested* fields as well. If its value is an iterable, only missing fields listed in that iterable will be ignored. Use dot delimiters to specify nested fields.

Returns A dictionary of validation errors.

New in version 1.1.0.

validate_email(*email: str*)

validate_id(*user_id: int*)

```
class app.serializers.user.VerifyRoleId(*, load_default: Any = <marshmallow.missing>, missing: Any = <marshmallow.missing>, dump_default: Any = <marshmallow.missing>, default: Any = <marshmallow.missing>, data_key: Optional[str] = None, attribute: Optional[str] = None, validate: Optional[Union[Callable[[Any], Any], Iterable[Callable[[Any], Any]]]] = None, required: bool = False, allow_none: Optional[bool] = None, load_only: bool = False, dump_only: bool = False, error_messages: Optional[Dict[str, str]] = None, metadata: Optional[Mapping[str, Any]] = None, **additional_metadata)
```

_CHECK_ATTRIBUTE = True

_bind_to_schema(*field_name, schema*)

Update field with values from its parent schema. Called by `Schema._bind_field`.

Parameters

- **field_name** (*str*) – Field name set in schema.
- **schema** (*Schema / Field*) – Parent object.

_creation_index = 129

_deserialize(*value, *args, **kwargs*)

Deserialize value. Concrete `Field` classes should implement this method.

Parameters

- **value** – The value to be deserialized.
- **attr** – The attribute/key in *data* to be deserialized.
- **data** – The raw input data passed to the `Schema.load`.
- **kwargs** – Field-specific keyword arguments.

Raises `ValidationError` – In case of formatting or validation failure.

Returns The deserialized value.

Changed in version 2.0.0: Added `attr` and `data` parameters.

Changed in version 3.0.0: Added `**kwargs` to signature.

_serialize(*value: Any, attr: str, obj: Any, **kwargs*)

Serializes *value* to a basic Python datatype. Noop by default. Concrete `Field` classes should implement this method.

Example:

```
class TitleCase(Field):
    def _serialize(self, value, attr, obj, **kwargs):
        if not value:
```

(continues on next page)

(continued from previous page)

```

    return ''
    return str(value).title()

```

Parameters

- **value** – The value to be serialized.
- **attr** (*str*) – The attribute or key on the object to be serialized.
- **obj** (*object*) – The object the value was pulled from.
- **kwargs** (*dict*) – Field-specific keyword arguments.

Returns The serialized value**_validate**(*value*)Perform validation on *value*. Raise a *ValidationError* if validation does not succeed.**property** **_validate_all****_validate_missing**(*value*)Validate missing values. Raise a *ValidationError* if *value* should be considered missing.**property** **context**

The context dictionary for the parent Schema.

property **default**

default_error_messages = {'null': 'Field may not be null.', 'required': 'Missing data for required field.', 'validator_failed': 'Invalid value.'}

Default error messages for various kinds of errors. The keys in this dictionary are passed to *Field.make_error*. The values are error messages passed to *marshmallow.exceptions.ValidationError*.

deserialize(*value: Any, attr: Optional[str] = None, data: Optional[Mapping[str, Any]] = None, **kwargs*)

Deserialize value.

Parameters

- **value** – The value to deserialize.
- **attr** – The attribute/key in *data* to deserialize.
- **data** – The raw input data passed to *Schema.load*.
- **kwargs** – Field-specific keyword arguments.

Raises **ValidationError** – If an invalid value is passed or if a required value is missing.**fail**(*key: str, **kwargs*)Helper method that raises a *ValidationError* with an error message from *self.error_messages*.Deprecated since version 3.0.0: Use *make_error* <*marshmallow.fields.Field.make_error*> instead.**get_value**(*obj, attr, accessor=None, default=<marshmallow.missing>*)

Return the value for a given key from an object.

Parameters

- **obj** (*object*) – The object to get the value from.
- **attr** (*str*) – The attribute/key in *obj* to get the value from.

- **accessor** (*callable*) – A callable used to retrieve the value of *attr* from the object *obj*. Defaults to *marshmallow.utils.get_value*.

make_error(*key: str, **kwargs*) → *marshmallow.exceptions.ValidationError*

Helper method to make a *ValidationError* with an error message from *self.error_messages*.

property missing

name = *None*

parent = *None*

root = *None*

serialize(*attr: str, obj: Any, accessor: Optional[Callable[[Any, str, Any], Any]] = None, **kwargs*)

Pulls the value for the given key from the object, applies the field's formatting and returns the result.

Parameters

- **attr** – The attribute/key to get from the object.
- **obj** – The object to access the attribute/key from.
- **accessor** – Function used to access values from *obj*.
- **kwargs** – Field-specific keyword arguments.

2.1.9 app.services

Description

Registers services for managing business logic.

Modules

app.services.auth

app.services.base

app.services.document

app.services.role

app.services.task

app.services.user

app.services.auth

Description

Classes

AuthService()

app.services.auth.AuthService

class app.services.auth.AuthService

Bases: object

Methods

AuthService.__init__()

AuthService.check_token_status(token)

AuthService.
confirm_request_reset_password(...)

*AuthService.login_user(**kwargs)*

AuthService.logout_user()

AuthService.
*request_reset_password(**kwargs)*

app.services.auth.AuthService.__init__

AuthService.__init__()

app.services.auth.AuthService.check_token_status

AuthService.**check_token_status**(token)

app.services.auth.AuthService.confirm_request_reset_password

AuthService.**confirm_request_reset_password**(token: str, password: str) → str

app.services.auth.AuthService.login_user

`AuthService.login_user(**kwargs) → str`

app.services.auth.AuthService.logout_user

`static AuthService.logout_user()`

app.services.auth.AuthService.request_reset_password

`AuthService.request_reset_password(**kwargs)`

class app.services.auth.AuthService

`check_token_status(token)`

`confirm_request_reset_password(token: str, password: str) → str`

`login_user(**kwargs) → str`

`static logout_user()`

`request_reset_password(**kwargs)`

app.services.base

Description

Classes

*BaseService(*args, **kwargs)*

app.services.base.BaseService

class app.services.base.BaseService(*args, **kwargs)

Bases: object

Methods

*BaseService.__init__(*args, **kwargs)*

*BaseService.create(**kwargs)*

BaseService.delete(record_id)

*BaseService.find(record_id, *args)*

continues on next page

Table 129 – continued from previous page

`BaseService.get(**kwargs)`

`BaseService.save(record_id, **kwargs)`

app.services.base.BaseService.__init__

`BaseService.__init__(*args, **kwargs)`

app.services.base.BaseService.create

`BaseService.create(**kwargs)`

app.services.base.BaseService.delete

`BaseService.delete(record_id: int)`

app.services.base.BaseService.find

`BaseService.find(record_id: int, *args)`

app.services.base.BaseService.get

`BaseService.get(**kwargs)`

app.services.base.BaseService.save

`BaseService.save(record_id: int, **kwargs)`

class app.services.base.BaseService(*args, **kwargs)

create(**kwargs)

delete(record_id: int)

find(record_id: int, *args)

get(**kwargs)

save(record_id: int, **kwargs)

app.services.document

Description

Classes

DocumentService()

app.services.document.DocumentService

class app.services.document.DocumentService
Bases: *app.services.base.BaseService*

Methods

DocumentService.__init__()

*DocumentService.create(**kwargs)*

DocumentService.delete(document_id)

DocumentService.find(document_id,
**args)*

*DocumentService.get(**kwargs)*

DocumentService.
get_document_content(...)

DocumentService.save(document_id,
***kwargs)*

app.services.document.DocumentService.__init__

DocumentService.__init__()

app.services.document.DocumentService.create

DocumentService.**create**(**kwargs)

app.services.document.DocumentService.delete

DocumentService.**delete**(document_id: int)

app.services.document.DocumentService.find

DocumentService.**find**(document_id: int, *args)

app.services.document.DocumentService.get

DocumentService.**get**(**kwargs)

app.services.document.DocumentService.get_document_content

DocumentService.**get_document_content**(document_id: int, **kwargs)

app.services.document.DocumentService.save

DocumentService.**save**(document_id: int, **kwargs)

class app.services.document.DocumentService

create(**kwargs)

delete(document_id: int)

find(document_id: int, *args)

get(**kwargs)

get_document_content(document_id: int, **kwargs)

save(document_id: int, **kwargs)

app.services.role**Description****Classes**

RoleService()

app.services.role.RoleService

```
class app.services.role.RoleService
    Bases: app.services.base.BaseService
```

Methods

RoleService.__init__()

*RoleService.create(**kwargs)*

RoleService.delete(role_id)

*RoleService.find(role_id, *args)*

*RoleService.get(**kwargs)*

*RoleService.save(role_id, **kwargs)*

app.services.role.RoleService.__init__

RoleService.__init__()

app.services.role.RoleService.create

RoleService.create(**kwargs)

app.services.role.RoleService.delete

RoleService.delete(role_id: int)

app.services.role.RoleService.find

RoleService.find(role_id: int, *args)

app.services.role.RoleService.get

RoleService.get(**kwargs)

app.services.role.RoleService.save

RoleService.**save**(role_id: int, **kwargs)

class app.services.role.RoleService

create(**kwargs)

delete(role_id: int)

find(role_id: int, *args)

get(**kwargs)

save(role_id: int, **kwargs)

app.services.task**Description****Classes**

TaskService()

app.services.task.TaskService

class app.services.task.TaskService

Bases: object

Methods

TaskService.__init__()

*TaskService.
check_task_status(task_id)*

*TaskService.
export_user_data_in_excel(data)*

*TaskService.
export_user_data_in_excel_and_word(...)*

*TaskService.
export_user_data_in_word(data, args)*

TaskService.find_by_id(task_id)

*TaskService.
reset_password_email(**kwargs)*

*TaskService.
send_create_user_email(**kwargs)*

app.services.task.TaskService.__init__

TaskService.__init__()

app.services.task.TaskService.check_task_status

TaskService.**check_task_status**(*task_id: str*) → dict

app.services.task.TaskService.export_user_data_in_excel

TaskService.**export_user_data_in_excel**(*data*)

app.services.task.TaskService.export_user_data_in_excel_and_word

TaskService.**export_user_data_in_excel_and_word**(*data, args*)

app.services.task.TaskService.export_user_data_in_word

TaskService.**export_user_data_in_word**(*data: dict, args: dict*)

app.services.task.TaskService.find_by_id

TaskService.**find_by_id**(*task_id: str*) → celery.local.PromiseProxy

app.services.task.TaskService.reset_password_email

TaskService.**reset_password_email**(***kwargs*)

app.services.task.TaskService.send_create_user_email

TaskService.**send_create_user_email**(***kwargs*)

class app.services.task.TaskService

check_task_status(*task_id: str*) → dict

export_user_data_in_excel(*data*)

export_user_data_in_excel_and_word(*data, args*)

export_user_data_in_word(*data: dict, args: dict*)

find_by_id(*task_id: str*) → celery.local.PromiseProxy

reset_password_email(***kwargs*)

send_create_user_email(***kwargs*)

app.services.user

Description

Classes

*UserService(*args, **kwargs)*

app.services.user.UserService

```
class app.services.user.UserService(*args, **kwargs)
    Bases: app.services.base.BaseService
```

Methods

*UserService.__init__(*args, **kwargs)*

UserService.create(user_data)

UserService.delete(user_id)

*UserService.find(user_id, *args)*

*UserService.get(**kwargs)*

*UserService.save(user_id, **kwargs)*

app.services.user.UserService.__init__

UserService.__init__(*args, **kwargs)

app.services.user.UserService.create

UserService.create(user_data)

app.services.user.UserService.delete

UserService.delete(user_id: int)

app.services.user.UserService.find

`UserService.find(user_id: int, *args)`

app.services.user.UserService.get

`UserService.get(**kwargs)`

app.services.user.UserService.save

`UserService.save(user_id: int, **kwargs)`

class app.services.user.UserService(*args, **kwargs)

create(user_data)

delete(user_id: int)

find(user_id: int, *args)

get(**kwargs)

save(user_id: int, **kwargs)

2.1.10 app.swagger

Description

Models registered in Swagger.

Modules

app.swagger.auth

app.swagger.core

app.swagger.document

app.swagger.role

app.swagger.user

app.swagger.auth

Description

app.swagger.core

Description

app.swagger.document

Description

app.swagger.role

Description

app.swagger.user

Description

2.1.11 app.utils

Description

Collection of functions and classes which make common patterns shorter and easier.

Modules

app.utils.constants

app.utils.decorators

app.utils.file_storage

app.utils.libreoffice

app.utils.request_query_operator

Module for creating a Peewee filter query via dynamic way.

app.utils.constants

Description

Attributes

MS_EXCEL_MIME_TYPE

REQUEST_QUERY_DELIMITER is used for converting requests field values to a list, for example: Request send these values: field_operator: contains field_values: valueA;valueB;valueC

app.utils.constants.MS_EXCEL_MIME_TYPE

app.utils.constants.MS_EXCEL_MIME_TYPE =

'application/vnd.openxmlformats-officedocument.spreadsheetml.sheet'

REQUEST_QUERY_DELIMITER is used for converting requests field values to a list, for example:

Request send these values: field_operator: contains field_values: valueA;valueB;valueC

The delimiter operator splits values to a list of values: field_values: [valueA, valueB, valueC]

app.utils.constants.MS_EXCEL_MIME_TYPE =

'application/vnd.openxmlformats-officedocument.spreadsheetml.sheet'

REQUEST_QUERY_DELIMITER is used for converting requests field values to a list, for example:

Request send these values: field_operator: contains field_values: valueA;valueB;valueC

The delimiter operator splits values to a list of values: field_values: [valueA, valueB, valueC]

app.utils.decorators

Description

Functions

token_required(fnc)

app.utils.decorators.token_required

app.utils.decorators.token_required(*fnc*)

app.utils.decorators.token_required(*fnc*)

app.utils.file_storage

Description

Classes

FileStorage()

app.utils.file_storage.FileStorage**class** app.utils.file_storage.**FileStorage**

Bases: object

Methods*FileStorage.__init__()**FileStorage.copy_file*(src, dst)*FileStorage.get_basename*(filename[, ...])*FileStorage.get_filesize*(filename)*FileStorage.rename*(src, dst)*FileStorage.save_bytes*(file_content, filename)**app.utils.file_storage.FileStorage.__init__**

FileStorage.__init__()

app.utils.file_storage.FileStorage.copy_file**static** FileStorage.**copy_file**(src: str, dst: str) → None**app.utils.file_storage.FileStorage.get_basename****static** FileStorage.**get_basename**(filename: str, include_path: bool = False) → str**app.utils.file_storage.FileStorage.get_filesize****static** FileStorage.**get_filesize**(filename: str) → int**app.utils.file_storage.FileStorage.rename****static** FileStorage.**rename**(src: str, dst: str) → None

app.utils.file_storage.FileStorage.save_bytes

FileStorage.**save_bytes**(*file_content: bytes, filename: str, override: bool = False*)

class app.utils.file_storage.**FileStorage**

static **copy_file**(*src: str, dst: str*) → None

static **get_basename**(*filename: str, include_path: bool = False*) → str

static **get_filesize**(*filename: str*) → int

static **rename**(*src: str, dst: str*) → None

save_bytes(*file_content: bytes, filename: str, override: bool = False*)

app.utils.libreoffice**Description****Functions**

convert_to(folder, source)

libreoffice_exec()

app.utils.libreoffice.convert_to

app.utils.libreoffice.**convert_to**(*folder: str, source: str*) → str

app.utils.libreoffice.libreoffice_exec

app.utils.libreoffice.**libreoffice_exec**() → str

Exceptions

LibreOfficeError(output)

app.utils.libreoffice.LibreOfficeError

exception app.utils.libreoffice.LibreOfficeError(*output*)

exception app.utils.libreoffice.LibreOfficeError(*output*)

args

with_traceback()

Exception.with_traceback(tb) – set self.__traceback__ to tb and return self.

app.utils.libreoffice.**convert_to**(*folder: str, source: str*) → str

app.utils.libreoffice.**libreoffice_exec**() → str

app.utils.request_query_operator**Description**

Module for creating a Peewee filter query via dynamic way.

Next module is used for creating a Peewee query based on request fields.

References

Query operators http://docs.peewee-orm.com/en/latest/peewee/query_operators.html

Classes

Helper()

RequestQueryOperator()

app.utils.request_query_operator.Helper

class app.utils.request_query_operator.Helper

Bases: object

Methods

Helper.__init__()

Helper.build_clause_operators(field,
...)

Helper.build_order_by(db_model, re- Build sorting fields with zero or more Column-
quest_data) like objects to order by.

Helper.build_sql_expression(field, ...)

continues on next page

Table 147 – continued from previous page

<i>Helper.build_string_clause</i> (field, ...)	Build string clauses.
--	-----------------------

app.utils.request_query_operator.Helper.__init__

```
Helper.__init__()
```

app.utils.request_query_operator.Helper.build_clause_operators

Helper.build_clause_operators(*field: peewee.Field, field_operator: str, field_value*) → tuple

app.utils.request_query_operator.Helper.build_order_by

static `Helper.build_order_by(db_model: Type[peewee.Model], request_data: dict) → list`
Build sorting fields with zero or more Column-like objects to order by.

Example

Peewee query: `User.select().order_by(User.created_at.asc())`

```
Request fields: >>> from app.models.user import User >>> db_model = User
>>> request_data = {'order': [{'sorting': 'asc', 'field_name': 'created_at'}]} >>>
Helper.build_order_by(db_model, request_data) [<peewee.Ordering object at ...>]
```

Notes

Actually is not possible to order across joins.

References

<http://docs.peewee-orm.com/en/latest/peewee/querying.html#sorting-records> http://docs.peewee-orm.com/en/latest/peewee/api.html#Query.order_by

app.utils.request_query_operator.Helper.build_sql_expression

Helper.build_sql_expression(*field*: peewee.Field, *field_operator*: str, *field_value*)

app.utils.request_query_operator.Helper.build_string_clause

Helper.build_string_clause(*field*: *peewee.Field*, *field_operator*: *str*, *field_value*) → tuple
Build string clauses.

You can find next string operators:	+-----+	Name
Description +=====+		
eq x equals y +-----+		ne x is not equal to y
+-----+		contains Wild-card search for substring
+-----+		ncontains Wild-card not search for substring
+-----+		startswith Search for values beginning with

prefix | +-----+-----+ | endswith | Search for values ending
with suffix | +-----+-----+

Example

TODO: Pending to define

app.utils.request_query_operator.RequestQueryOperator

class app.utils.request_query_operator.RequestQueryOperator
Bases: object

Methods

RequestQueryOperator.__init__()

*RequestQueryOperator.
create_search_query(...)*

*RequestQueryOperator.
get_request_query_fields(...)*

app.utils.request_query_operator.RequestQueryOperator.__init__

RequestQueryOperator.__init__()

app.utils.request_query_operator.RequestQueryOperator.create_search_query

static RequestQueryOperator.create_search_query(*db_model: Type[peewee.Model],
query: peewee.ModelSelect, data: Optional[dict] = None*) →
peewee.ModelSelect

app.utils.request_query_operator.RequestQueryOperator.get_request_query_fields

static RequestQueryOperator.get_request_query_fields(*db_model: Type[peewee.Model],
request_data=None*) → tuple

class app.utils.request_query_operator.Helper

build_clause_operators(*field: peewee.Field, field_operator: str, field_value*) → tuple

static build_order_by(*db_model: Type[peewee.Model], request_data: dict*) → list
Build sorting fields with zero or more Column-like objects to order by.

Example

Peewee query: `User.select().order_by(User.created_at.asc())`

```
Request fields: >>> from app.models.user import User >>> db_model = User >>> request_data = {'order': [{'sorting': 'asc', 'field_name': 'created_at'}]} >>> Helper.build_order_by(db_model, request_data)
[<peewee.Ordering object at ...>]
```

Notes

Actually is not possible to order across joins.

References

<http://docs.peewee-orm.com/en/latest/peewee/querying.html#sorting-records> http://docs.peewee-orm.com/en/latest/peewee/api.html#Query.order_by

build_sql_expression(*field*: *peewee.Field*, *field_operator*: *str*, *field_value*)**build_string_clause**(*field*: *peewee.Field*, *field_operator*: *str*, *field_value*) → tuple

Build string clauses.

You can find next string operators:						
	+	-----	+	-----	+	Name
Description	+	=====	+	=====	+	
eq x equals y	+	-----	+	-----	+	ne x is not equal to y
+	+	-----	+	-----	+	contains Wild-card search for substring
+	+	-----	+	-----	+	ncontains Wild-card not search for substring
+	+	-----	+	-----	+	startswith Search for values beginning with prefix
+-----	+	-----	+	-----	+	endswith Search for values ending with suffix
+	+	-----	+	-----	+	

Example

TODO: Pending to define

```
class app.utils.request_query_operator.RequestQueryOperator
```

```
static create_search_query(db_model: Type[peewee.Model], query: peewee.ModelSelect, data:
    Optional[dict] = None) → peewee.ModelSelect
```

```
static get_request_query_fields(db_model: Type[peewee.Model], request_data=None) → tuple
```

Functions

```
filter_by_keys(data, keys)
```

```
find_longest_word(word_list)
```

`get_attr_from_module(module, attr)`

Get attribute from a module.

```
get_request_file([field_name])
```

continues on next page

Table 149 – continued from previous page

`ignore_keys`(data, exclude)

`pos_to_char`(pos)

`to_readable`(obj)

`app.utils.filter_by_keys`

`app.utils.filter_by_keys`(data: dict, keys: list) → dict

`app.utils.find_longest_word`

`app.utils.find_longest_word`(word_list: list) → str

`app.utils.get_attr_from_module`

`app.utils.get_attr_from_module`(module: str, attr: str) → any

Get attribute from a module.

Parameters

- **module** (str) – Module absolute path.
- **attr** (str) – Module’s attribute. It could be any kind of variable belongs to module.

Examples

```
>>> from app.utils import get_attr_from_module
>>> module_path = 'app.blueprints.base'
>>> module_attr = 'blueprint'
>>> get_attr_from_module(module_path, module_attr)
<flask.blueprints.Blueprint object at ...>
```

Raises

- **ImportError** – Module doesn’t exist.
- **AttributeError** – Attribute doesn’t exist in module.

`app.utils.get_request_file`

`app.utils.get_request_file`(field_name: Optional[str] = None) → dict

app.utils.ignore_keys

`app.utils.ignore_keys(data: dict, exclude: list) → dict`

app.utils.pos_to_char

`app.utils.pos_to_char(pos: int) → str`

app.utils.to_readable

`app.utils.to_readable(obj: object) → object`

`app.utils.filter_by_keys(data: dict, keys: list) → dict`

`app.utils.find_longest_word(word_list: list) → str`

`app.utils.get_attr_from_module(module: str, attr: str) → any`
Get attribute from a module.

Parameters

- **module** (*str*) – Module absolute path.
- **attr** (*str*) – Module's attribute. It could be any kind of variable belongs to module.

Examples

```
>>> from app.utils import get_attr_from_module
>>> module_path = 'app.blueprints.base'
>>> module_attr = 'blueprint'
>>> get_attr_from_module(module_path, module_attr)
<flask.blueprints.Blueprint object at ...>
```

Raises

- **ImportError** – Module doesn't exist.
- **AttributeError** – Attribute doesn't exist in module.

`app.utils.get_request_file(field_name: Optional[str] = None) → dict`

`app.utils.ignore_keys(data: dict, exclude: list) → dict`

`app.utils.pos_to_char(pos: int) → str`

`app.utils.to_readable(obj: object) → object`

Functions

<code>create_app(env_config)</code>	Builds an application based on environment configuration.
-------------------------------------	---

2.1.12 app.create_app

`app.create_app(env_config: str) → flask.app.Flask`

Builds an application based on environment configuration.

Parameters `env_config` – Environment configuration.

Returns A *flask.flask* instance.

Return type Flask

Notes

Environment configuration values could be:

```
config.ProdConfig
config.DevConfig
config.TestConfig
```

`app._init_logging(app: flask.app.Flask) → None`

`app._register_blueprints(app: flask.app.Flask) → None`

`app.create_app(env_config: str) → flask.app.Flask`

Builds an application based on environment configuration.

Parameters `env_config` – Environment configuration.

Returns A *flask.flask* instance.

Return type Flask

Notes

Environment configuration values could be:

```
config.ProdConfig
config.DevConfig
config.TestConfig
```

2.2 database

Description

Package for managing the database.

The database package can creates and migrates tables and it can fills them with fake data.

Modules

<code>database.factories</code>	Package contains factories modules.
<code>database.migrations</code>	
<code>database.seeds</code>	

2.2.1 database.factories

Description

Package contains factories modules.

A factory is a database model filled with fake data. The factory purposes is creating records in a simple way.

The module is used in testing and seeds.

References

The factory concept is based on [Laravel factories](#)

Classes

<code>Factory(model_name[, records])</code>	Class for managing factories based on database models.
---	--

database.factories.Factory

class database.factories.Factory(*model_name: str, records: int = 1*)

Bases: object

Class for managing factories based on database models.

Create and save instances of database models or dicts based on database models registered in the application.

make(*self, params: dict = None, to_dict: bool = False, exclude: list = None*)

Create instances of database models with fake data.

save(*self, params: dict = None*)

Save instances of database models in the database.

Examples

How to create a fake user without save in database from command line:

```
source venv/bin/activate
flask shell
>>> user_factory = Factory('User')
>>> user = user_factory.make() # An instance of database model
<User: None>
>>> user.__data__ # You can see user data on this way
```

Oh, Wait!

```
>>> from pprint import pprint
>>> pprint(user.__data__) # Even better!
```

You can save the user in the database.

```
>>> user.save()
1
```

Factory can create a dictionary instead of an instance of database model.

```
>>> user = user_factory.make(to_dict=True)
>>> pprint(user)
```

Also can set params too.

```
>>> user_factory = Factory('User')
>>> user = user_factory.make({'name': 'Ruben', 'last_name': 'Rodriguez'})
>>> user.name
'Ruben'
>>> user.last_name
'Rodriguez'
```

Factory allow to make many users in once time.

```
>>> user_factory = Factory('User', 3)
>>> users = user_factory.make()
[<User: None>, <User: None>, <User: None>]
```

If you want to fill some params later then you can pass a fieldnames list to the factory of thats fields that you don't want to fill yet.

```
>>> user_factory = Factory('User')
>>> user = user_factory.make(exclude=['name', 'birth_date'])
>>> user.name
None
>>> user.birth_date
None
```

If you only need to save data you can do it.

```
>>> Factory('User', 3).save()
[<User: 1>, <User: 2>, <User: 3>]
```

And you can set params for all users.

```
>>> Factory('User', 3).save({'name': 'Ruben'})
[<User: 4>, <User: 5>, <User: 6>]
```

Methods

<code>Factory.__init__(model_name[, records])</code>	Register as many factories as given records.
<code>Factory.make([params, to_dict, exclude])</code>	Create instances of database model with fake data.
<code>Factory.save([params])</code>	Save instances of database model in the database.

database.factories.Factory.__init__

`Factory.__init__(model_name: str, records: int = 1)`
Register as many factories as given records.

database.factories.Factory.make

`Factory.make(params: Optional[dict] = None, to_dict: bool = False, exclude: Optional[list] = None) → any`

Create instances of database model with fake data.

Parameters

- **params** (*dict*) – Params to set when an instance of database model is created.
- **to_dict** (*bool*) – If is True returns a dict otherwise is an instance of database model. By default is False.
- **exclude** (*list*) – Params are not going to be filled. These fields are equals to None.

Returns Could be a dict, a list or an instance of database model.

Return type any

database.factories.Factory.save

`Factory.save(params: Optional[dict] = None) → any`

Save instances of database model in the database.

Parameters **params** (*dict*) – Params to set when an instance of database model is created.

Returns Could be a list or an instance of database model.

Return type any

class database.factories.**Factory**(*model_name: str, records: int = 1*)

Class for managing factories based on database models.

Create and save instances of database models or dicts based on database models registered in the application.

make(*self, params: dict = None, to_dict: bool = False, exclude: list = None*)

Create instances of database models with fake data.

save(self, params: dict = None)
 Save instances of database models in the database.

Examples

How to create a fake user without save in database from command line:

```
source venv/bin/activate
flask shell
>>> user_factory = Factory('User')
>>> user = user_factory.make() # An instance of database model
<User: None>
>>> user.__data__ # You can see user data on this way
```

Oh, Wait!

```
>>> from pprint import pprint
>>> pprint(user.__data__) # Even better!
```

You can save the user in the database.

```
>>> user.save()
1
```

Factory can create a dictionary instead of an instance of database model.

```
>>> user = user_factory.make(to_dict=True)
>>> pprint(user)
```

Also can set params too.

```
>>> user_factory = Factory('User')
>>> user = user_factory.make({'name': 'Ruben', 'last_name': 'Rodriguez'})
>>> user.name
'Ruben'
>>> user.last_name
'Rodriguez'
```

Factory allow to make many users in once time.

```
>>> user_factory = Factory('User', 3)
>>> users = user_factory.make()
[<User: None>, <User: None>, <User: None>]
```

If you want to fill some params later then you can pass a fieldnames list to the factory of thats fields that you don't want to fill yet.

```
>>> user_factory = Factory('User')
>>> user = user_factory.make(exclude=['name', 'birth_date'])
>>> user.name
None
>>> user.birth_date
None
```

If you only need to save data you can do it.

```
>>> Factory('User', 3).save()
[<User: 1>, <User: 2>, <User: 3>]
```

And you can set params for all users.

```
>>> Factory('User', 3).save({'name': 'Ruben'})
[<User: 4>, <User: 5>, <User: 6>]
```

__check_exists_factory(*factory_classname: str, model_name: str*) → None

__current_module = 'database.factories'

__models: list = None

make(*params: Optional[dict] = None, to_dict: bool = False, exclude: Optional[list] = None*) → any
Create instances of database model with fake data.

Parameters

- **params** (*dict*) – Params to set when an instance of database model is created.
- **to_dict** (*bool*) – If is True returns a dict otherwise is an instance of database model. By default is False.
- **exclude** (*list*) – Params are not going to be filled. These fields are equals to None.

Returns Could be a dict, a list or an instance of database model.

Return type any

save(*params: Optional[dict] = None*) → any
Save instances of database model in the database.

Parameters **params** (*dict*) – Params to set when an instance of database model is created.

Returns Could be a list or an instance of database model.

Return type any

2.2.2 database.migrations

Description

Modules

```
database.migrations.  
aaa_add_genre_column_on_user_table  
database.migrations.  
aab_add_created_by_column_on_user_table  
database.migrations.  
aac_create_documents_table  
database.migrations.  
aad_create_user_roles_table  
database.migrations.  
aaf_remove_role_slug_column  
database.migrations.  
aag_add_fs_uniquifier_column_on_users_table
```

database.migrations.aaa_add_genre_column_on_user_table

Description

Classes

AddGenreColumnOnUserTable()

database.migrations.aaa_add_genre_column_on_user_table.AddGenreColumnOnUserTable

class database.migrations.aaa_add_genre_column_on_user_table.**AddGenreColumnOnUserTable**
 Bases: object

Methods

AddGenreColumnOnUserTable.__init__()

AddGenreColumnOnUserTable.down()

AddGenreColumnOnUserTable.up()

database.migrations.aaa_add_genre_column_on_user_table.AddGenreColumnOnUserTable.__init__

AddGenreColumnOnUserTable.**__init__()**

database.migrations.aaa_add_genre_column_on_user_table.AddGenreColumnOnUserTable.down

AddGenreColumnOnUserTable.**down()**

database.migrations.aaa_add_genre_column_on_user_table.AddGenreColumnOnUserTable.up

AddGenreColumnOnUserTable.**up()**

class database.migrations.aaa_add_genre_column_on_user_table.**AddGenreColumnOnUserTable**

_exists_column() → bool

down()

up()

database.migrations.aab_add_created_by_column_on_user_table

Description

Classes

AddCreatedByColumnOnUserTable()

database.migrations.aab_add_created_by_column_on_user_table.AddCreatedByColumnOnUserTable

class

database.migrations.aab_add_created_by_column_on_user_table.AddCreatedByColumnOnUserTable
Bases: object

Methods

AddCreatedByColumnOnUserTable.

__init__()

AddCreatedByColumnOnUserTable.down()

AddCreatedByColumnOnUserTable.up()

database.migrations.aab_add_created_by_column_on_user_table.AddCreatedByColumnOnUserTable.

AddCreatedByColumnOnUserTable.__init__()

database.migrations.aab_add_created_by_column_on_user_table.AddCreatedByColumnOnUserTable.d

AddCreatedByColumnOnUserTable.down()

database.migrations.aab_add_created_by_column_on_user_table.AddCreatedByColumnOnUserTable.up

AddCreatedByColumnOnUserTable.up()

class

database.migrations.aab_add_created_by_column_on_user_table.AddCreatedByColumnOnUserTable

_exists_column() → bool

down()

up()

database.migrations.aac_create_documents_table

Description

Classes

CreateDocumentsTable()

database.migrations.aac_create_documents_table.CreateDocumentsTable

class database.migrations.aac_create_documents_table.**CreateDocumentsTable**

Bases: object

Methods

CreateDocumentsTable.__init__()

CreateDocumentsTable.down()

CreateDocumentsTable.up()

database.migrations.aac_create_documents_table.CreateDocumentsTable.__init__

CreateDocumentsTable.**__init__**()

database.migrations.aac_create_documents_table.CreateDocumentsTable.down

CreateDocumentsTable.**down**()

database.migrations.aac_create_documents_table.CreateDocumentsTable.up

CreateDocumentsTable.**up**()

class database.migrations.aac_create_documents_table.**CreateDocumentsTable**

_exists_table() → bool

down()

up()

database.migrations.aad_create_user_roles_table

Description

Classes

CreateUserRolesTable()

database.migrations.aad_create_user_roles_table.CreateUserRolesTable

class database.migrations.aad_create_user_roles_table.CreateUserRolesTable

Bases: object

Methods

CreateUserRolesTable.__init__()

CreateUserRolesTable.down()

CreateUserRolesTable.up()

database.migrations.aad_create_user_roles_table.CreateUserRolesTable.__init__

CreateUserRolesTable.__init__()

database.migrations.aad_create_user_roles_table.CreateUserRolesTable.down

CreateUserRolesTable.down()

database.migrations.aad_create_user_roles_table.CreateUserRolesTable.up

CreateUserRolesTable.up()

class database.migrations.aad_create_user_roles_table.CreateUserRolesTable

static **_add_foreign_key_constraint_users_table()** → None

https://www.sqlite.org/lang_altertable.html

static **_drop_foreign_key_constraint_users_table()** → list

https://www.sqlite.org/lang_altertable.html

_exists_table() → bool

down()

up()


```

class database.migrations.aad_create_user_roles_table._OldUser(*args, **kwargs)

    DoesNotExist
        alias of database.migrations.aad_create_user_roles_table._OldUserDoesNotExist
    _coerce = True
    _meta = <peewee.Metadata object>
    classmethod _normalize_data(data, kwargs)
    property _pk
    _pk_expr()
    _populate_unsaved_relations(field_dict)
    _prune_fields(field_dict, only)
    _schema = <peewee.SchemaManager object>
    active = <BooleanField: _OldUser.active>
    classmethod add_index(*fields, **kwargs)
    classmethod alias(alias=None)
    classmethod bind(database, bind_refs=True, bind_backrefs=True, _exclude=None)
    classmethod bind_ctx(database, bind_refs=True, bind_backrefs=True)
    birth_date = <DateField: _OldUser.birth_date>
    classmethod bulk_create(model_list, batch_size=None)
    classmethod bulk_update(model_list, fields, batch_size=None)
    children
    clone()
    coerce(_coerce=True)
    static copy(method)
    classmethod create(**query)
    classmethod create_table(safe=True, **options)
    created_at = <TimestampField: _OldUser.created_at>
    created_by = <ForeignKeyField: _OldUser.created_by>
    created_by_id = <ForeignKeyField: _OldUser.created_by>
    classmethod delete()
    classmethod delete_by_id(pk)
    delete_instance(recursive=False, delete_nullable=False)
    deleted_at = <TimestampField: _OldUser.deleted_at>
    dependencies(search_nullable=False)
    property dirty_fields
    classmethod drop_table(safe=True, drop_sequences=True, **options)
    email = <CharField: _OldUser.email>

```

```
classmethod filter(*dq_nodes, **filters)
genre = <FixedCharField: _OldUser.genre>
classmethod get(*query, **filters)
classmethod get_by_id(pk)
classmethod get_fields(exclude: Optional[list] = None, include: Optional[list] = None, sort_order:
                        Optional[list] = None) → set

get_id()
classmethod get_or_create(**kwargs)
classmethod get_or_none(*query, **filters)
id = <AutoField: _OldUser.id>
classmethod index(*fields, **kwargs)
classmethod insert(_Model__data=None, **insert)
classmethod insert_from(query, fields)
classmethod insert_many(rows, fields=None)
property is_active
is_alias()
property is_anonymous
property is_authenticated
is_dirty()
last_name = <CharField: _OldUser.last_name>
name = <CharField: _OldUser.name>
classmethod noop()
password = <CharField: _OldUser.password>
static raw(query: str)
reload()
classmethod replace(_Model__data=None, **insert)
classmethod replace_many(rows, fields=None)
role = <ForeignKeyField: _OldUser.role>
role_id = <ForeignKeyField: _OldUser.role>
abstract save(*args: list, **kwargs: dict) → int
classmethod select(*fields)
classmethod set_by_id(key, value)
classmethod table_exists()
classmethod truncate_table(**options)
unwrap()
classmethod update(_Model__data=None, **update)
```

```
updated_at = <TimestampField: _OldUser.updated_at>
classmethod validate_model()
```

database.migrations.aaf_remove_role_slug_column

Description

Classes

RemoveRoleSlugColumn()

database.migrations.aaf_remove_role_slug_column.RemoveRoleSlugColumn

```
class database.migrations.aaf_remove_role_slug_column.RemoveRoleSlugColumn
    Bases: object
```

Methods

RemoveRoleSlugColumn.__init__()

RemoveRoleSlugColumn.down()

RemoveRoleSlugColumn.up()

database.migrations.aaf_remove_role_slug_column.RemoveRoleSlugColumn.__init__

RemoveRoleSlugColumn.__init__()

database.migrations.aaf_remove_role_slug_column.RemoveRoleSlugColumn.down

RemoveRoleSlugColumn.down()

database.migrations.aaf_remove_role_slug_column.RemoveRoleSlugColumn.up

RemoveRoleSlugColumn.up()

```
class database.migrations.aaf_remove_role_slug_column.RemoveRoleSlugColumn
```

```
    static _add_unique_constraint_roles_table() → None
        https://www.sqlite.org/lang\_altertable.html
```

```
    static _drop_unique_constraint_roles_table() → None
        https://www.sqlite.org/lang\_altertable.html
```

```
    _exists_column() → bool
```

```
down()
up()
class database.migrations.aaf_remove_role_slug_column._OldRole(*args, **kwargs)

DoesNotExist
    alias of database.migrations.aaf_remove_role_slug_column._OldRoleDoesNotExist
_coerce = True
_meta = <peewee.Metadata object>
classmethod _normalize_data(data, kwargs)
property _pk
_pk_expr()
_populate_unsaved_relations(field_dict)
_prune_fields(field_dict, only)
_schema = <peewee.SchemaManager object>
classmethod add_index(*fields, **kwargs)
add_permissions(permissions: Union[set, list, str]) → None
    Add one or more permissions to role.

    Parameters permissions – a set, list, or single string.

    New in version 3.3.0.

    Deprecated since version 3.4.4: Use UserDatastore.add_permissions_to_role()
classmethod alias(alias=None)
classmethod bind(database, bind_refs=True, bind_backrefs=True, _exclude=None)
classmethod bind_ctx(database, bind_refs=True, bind_backrefs=True)
classmethod bulk_create(model_list, batch_size=None)
classmethod bulk_update(model_list, fields, batch_size=None)
clone()
coerce(_coerce=True)
static copy(method)
classmethod create(**query)
classmethod create_table(safe=True, **options)
created_at = <TimestampField: _OldRole.created_at>
classmethod delete()
classmethod delete_by_id(pk)
delete_instance(recursive=False, delete_nullable=False)
deleted_at = <TimestampField: _OldRole.deleted_at>
dependencies(search_nullable=False)
description = <TextField: _OldRole.description>
```

```

property dirty_fields
classmethod drop_table(safe=True, drop_sequences=True, **options)
classmethod filter(*dq_nodes, **filters)
classmethod get(*query, **filters)
classmethod get_by_id(pk)
classmethod get_fields(exclude: Optional[list] = None, include: Optional[list] = None, sort_order:
                        Optional[list] = None) → set

get_id()
classmethod get_or_create(**kwargs)
classmethod get_or_none(*query, **filters)

get_permissions() → set
    Return set of permissions associated with role.

    Supports permissions being a comma separated string, an iterable, or a set based on how the underlying
    DB model was built.

    New in version 3.3.0.

id = <AutoField: _OldRole.id>
classmethod index(*fields, **kwargs)
classmethod insert(_Model__data=None, **insert)
classmethod insert_from(query, fields)
classmethod insert_many(rows, fields=None)

is_alias()
is_dirty()

name = <CharField: _OldRole.name>
classmethod noop()
static raw(query: str)
reload()

remove_permissions(permissions: Union[set, list, str]) → None
    Remove one or more permissions from role.

    Parameters permissions – a set, list, or single string.

    New in version 3.3.0.

    Deprecated since version 3.4.4: Use UserDatastore.remove_permissions_from_role()

classmethod replace(_Model__data=None, **insert)
classmethod replace_many(rows, fields=None)
abstract save(*args: list, **kwargs: dict) → int
classmethod select(*fields)
classmethod set_by_id(key, value)

slug = <CharField: _OldRole.slug>

```

```
classmethod table_exists()
classmethod truncate_table(**options)
unwrap()
classmethod update(_Model__data=None, **update)
updated_at = <TimestampField: _OldRole.updated_at>
classmethod validate_model()
```

database.migrations.aag_add_fs_uniquifier_column_on_users_table

Description

Classes

AddFsUniquifierColumnOnUsersTable()

database.migrations.aag_add_fs_uniquifier_column_on_users_table.AddFsUniquifierColumnOnUsersTable

```
class database.migrations.aag_add_fs_uniquifier_column_on_users_table.
AddFsUniquifierColumnOnUsersTable
    Bases: object
```

Methods

AddFsUniquifierColumnOnUsersTable.
__init__()

AddFsUniquifierColumnOnUsersTable.
down()

AddFsUniquifierColumnOnUsersTable.
up()

database.migrations.aag_add_fs_uniquifier_column_on_users_table.AddFsUniquifierColumnOnUsersTable

AddFsUniquifierColumnOnUsersTable.__init__()

```
database.migrations.aag_add_fs_uniquifier_column_on_users_table.AddFsUniquifierColumnOnUsersTable
```

```
AddFsUniquifierColumnOnUsersTable.down()
```

```
database.migrations.aag_add_fs_uniquifier_column_on_users_table.AddFsUniquifierColumnOnUsersTable
```

```
AddFsUniquifierColumnOnUsersTable.up()
```

```
class database.migrations.aag_add_fs_uniquifier_column_on_users_table.  
AddFsUniquifierColumnOnUsersTable
```

```
    _exists_column() → bool
```

```
    down()
```

```
    up()
```

Classes

```
Migration(*args, **kwargs)
```

database.migrations.Migration

```
class database.migrations.Migration(*args, **kwargs)  
    Bases: playhouse.flask_utils.FlaskDB.get_model_class.<locals>.BaseModel
```

Attributes

```
Migration.dirty_fields
```

```
Migration.id
```

```
Migration.name
```

```
database.migrations.Migration.dirty_fields
```

```
property Migration.dirty_fields
```

database.migrations.Migration.id

`Migration.id = <AutoField: BaseModel.id>`

database.migrations.Migration.name

`Migration.name = <CharField: Migration.name>`

Methods

Migration.__init__(*args, **kwargs)

Migration.add_index(*fields, **kwargs)

Migration.alias([alias])

Migration.bind(database[, bind_refs, ...])

Migration.bind_ctx(database[, bind_refs, ...])

Migration.bulk_create(model_list[, batch_size])

Migration.bulk_update(model_list, fields[, ...])

Migration.clone()

Migration.coerce([_coerce])

Migration.copy(method)

Migration.create(**query)

Migration.create_table([safe])

Migration.delete()

Migration.delete_by_id(pk)

Migration.delete_instance([recursive, ...])

Migration.dependencies([search_nullable])

Migration.drop_table([safe, drop_sequences])

Migration.filter(*dq_nodes, **filters)

Migration.get(*query, **filters)

Migration.get_by_id(pk)

continues on next page

Table 169 – continued from previous page

<i>Migration.get_id()</i>
<i>Migration.get_or_create(**kwargs)</i>
<i>Migration.get_or_none(*query, **filters)</i>
<i>Migration.index(*fields, **kwargs)</i>
<i>Migration.insert([_Model__data])</i>
<i>Migration.insert_from(query, fields)</i>
<i>Migration.insert_many(rows[, fields])</i>
<i>Migration.is_alias()</i>
<i>Migration.is_dirty()</i>
<i>Migration.noop()</i>
<i>Migration.raw(sql, *params)</i>
<i>Migration.replace([_Model__data])</i>
<i>Migration.replace_many(rows[, fields])</i>
<i>Migration.save([force_insert, only])</i>
<i>Migration.select(*fields)</i>
<i>Migration.set_by_id(key, value)</i>
<i>Migration.table_exists()</i>
<i>Migration.truncate_table(**options)</i>
<i>Migration.unwrap()</i>
<i>Migration.update([_Model__data])</i>
<i>Migration.validate_model()</i>

database.migrations.Migration.__init__

`Migration.__init__(*args, **kwargs)`

database.migrations.Migration.add_index

classmethod `Migration.add_index(*fields, **kwargs)`

database.migrations.Migration.alias

classmethod `Migration.alias(alias=None)`

database.migrations.Migration.bind

classmethod `Migration.bind(database, bind_refs=True, bind_backrefs=True, _exclude=None)`

database.migrations.Migration.bind_ctx

classmethod `Migration.bind_ctx(database, bind_refs=True, bind_backrefs=True)`

database.migrations.Migration.bulk_create

classmethod `Migration.bulk_create(model_list, batch_size=None)`

database.migrations.Migration.bulk_update

classmethod `Migration.bulk_update(model_list, fields, batch_size=None)`

database.migrations.Migration.clone

`Migration.clone()`

database.migrations.Migration.coerce

`Migration.coerce(_coerce=True)`

database.migrations.Migration.copy

static Migration.**copy**(*method*)

database.migrations.Migration.create

classmethod Migration.**create**(***query*)

database.migrations.Migration.create_table

classmethod Migration.**create_table**(*safe=True, **options*)

database.migrations.Migration.delete

classmethod Migration.**delete**()

database.migrations.Migration.delete_by_id

classmethod Migration.**delete_by_id**(*pk*)

database.migrations.Migration.delete_instance

Migration.**delete_instance**(*recursive=False, delete_nullable=False*)

database.migrations.Migration.dependencies

Migration.**dependencies**(*search_nullable=False*)

database.migrations.Migration.drop_table

classmethod Migration.**drop_table**(*safe=True, drop_sequences=True, **options*)

database.migrations.Migration.filter

classmethod Migration.**filter**(**dq_nodes, **filters*)

database.migrations.Migration.get

classmethod Migration.get(*query, **filters)

database.migrations.Migration.get_by_id

classmethod Migration.get_by_id(pk)

database.migrations.Migration.get_id

Migration.get_id()

database.migrations.Migration.get_or_create

classmethod Migration.get_or_create(**kwargs)

database.migrations.Migration.get_or_none

classmethod Migration.get_or_none(*query, **filters)

database.migrations.Migration.index

classmethod Migration.index(*fields, **kwargs)

database.migrations.Migration.insert

classmethod Migration.insert(_Model__data=None, **insert)

database.migrations.Migration.insert_from

classmethod Migration.insert_from(query, fields)

database.migrations.Migration.insert_many

classmethod Migration.insert_many(rows, fields=None)

database.migrations.Migration.is_alias

`Migration.is_alias()`

database.migrations.Migration.is_dirty

`Migration.is_dirty()`

database.migrations.Migration.noop

`classmethod Migration.noop()`

database.migrations.Migration.raw

`classmethod Migration.raw(sql, *params)`

database.migrations.Migration.replace

`classmethod Migration.replace(_Model__data=None, **insert)`

database.migrations.Migration.replace_many

`classmethod Migration.replace_many(rows, fields=None)`

database.migrations.Migration.save

`Migration.save(force_insert=False, only=None)`

database.migrations.Migration.select

`classmethod Migration.select(*fields)`

database.migrations.Migration.set_by_id

`classmethod Migration.set_by_id(key, value)`

database.migrations.Migration.table_exists

classmethod Migration.**table_exists()**

database.migrations.Migration.truncate_table

classmethod Migration.**truncate_table(**options)**

database.migrations.Migration.unwrap

Migration.**unwrap()**

database.migrations.Migration.update

classmethod Migration.**update(_Model__data=None, **update)**

database.migrations.Migration.validate_model

classmethod Migration.**validate_model()**

Functions

get_migration_names()

init_migrations([rollback])

migrate_actions(fnc)

rollback_actions(fnc)

database.migrations.get_migration_names

database.migrations.**get_migration_names()** → list

database.migrations.init_migrations

database.migrations.**init_migrations**(rollback: bool = False) → None

database.migrations.migrate_actions

database.migrations.**migrate_actions**(fnc)

database.migrations.rollback_actions

database.migrations.**rollback_actions**(fnc)

class database.migrations.**Migration**(*args, **kwargs)

DoesNotExist

alias of database.migrations.MigrationDoesNotExist

_coerce = True

_meta = <peewee.Metadata object>

classmethod **_normalize_data**(data, kwargs)

property **_pk**

_pk_expr()

_populate_unsaved_relations(field_dict)

_prune_fields(field_dict, only)

_schema = <peewee.SchemaManager object>

classmethod **add_index**(*fields, **kwargs)

classmethod **alias**(alias=None)

classmethod **bind**(database, bind_refs=True, bind_backrefs=True, _exclude=None)

classmethod **bind_ctx**(database, bind_refs=True, bind_backrefs=True)

classmethod **bulk_create**(model_list, batch_size=None)

classmethod **bulk_update**(model_list, fields, batch_size=None)

clone()

coerce(_coerce=True)

static **copy**(method)

classmethod **create**(**query)

classmethod **create_table**(safe=True, **options)

classmethod **delete**()

classmethod **delete_by_id**(pk)

delete_instance(recursive=False, delete_nullable=False)

dependencies(search_nullable=False)

property **dirty_fields**

```
classmethod drop_table(safe=True, drop_sequences=True, **options)
classmethod filter(*dq_nodes, **filters)
classmethod get(*query, **filters)
classmethod get_by_id(pk)
get_id()
classmethod get_or_create(**kwargs)
classmethod get_or_none(*query, **filters)
id = <AutoField: BaseModel.id>
classmethod index(*fields, **kwargs)
classmethod insert(_Model__data=None, **insert)
classmethod insert_from(query, fields)
classmethod insert_many(rows, fields=None)
is_alias()
is_dirty()
name = <CharField: Migration.name>
classmethod noop()
classmethod raw(sql, *params)
classmethod replace(_Model__data=None, **insert)
classmethod replace_many(rows, fields=None)
save(force_insert=False, only=None)
classmethod select(*fields)
classmethod set_by_id(key, value)
classmethod table_exists()
classmethod truncate_table(**options)
unwrap()
classmethod update(_Model__data=None, **update)
classmethod validate_model()

database.migrations.get_migration_names() → list
database.migrations.init_migrations(rollback: bool = False) → None
database.migrations.migrate_actions(fnc)
database.migrations.rollback_actions(fnc)
```


2.2.3 database.seeds

Description

Modules

database.seeds.document_seeder

database.seeds.role_seeder

database.seeds.user_seeder

database.seeds.document_seeder

Description

Classes

DocumentSeeder([rows])

database.seeds.document_seeder.DocumentSeeder

class database.seeds.document_seeder.**DocumentSeeder**(rows: int = 30)
Bases: object

Attributes

DocumentSeeder.name

database.seeds.document_seeder.DocumentSeeder.name

DocumentSeeder.name = 'DocumentSeeder'

Methods

DocumentSeeder.__init__([rows])

database.seeds.document_seeder.DocumentSeeder.__init__

DocumentSeeder.__init__(rows: int = 30)

class database.seeds.document_seeder.DocumentSeeder(rows: int = 30)

name = 'DocumentSeeder'

database.seeds.role_seeder

Description

Classes

RoleSeeder()

database.seeds.role_seeder.RoleSeeder

class database.seeds.role_seeder.RoleSeeder

Bases: object

Attributes

RoleSeeder.name

database.seeds.role_seeder.RoleSeeder.name

RoleSeeder.name = 'RoleSeeder'

Methods

RoleSeeder.__init__()

database.seeds.role_seeder.RoleSeeder.__init__

```
RoleSeeder.__init__()
```

```
class database.seeds.role_seeder.RoleSeeder
```

```
    static _create_admin_role() → None
```

```
    static _create_team_leader() → None
```

```
    static _create_worker_role() → None
```

```
    name = 'RoleSeeder'
```

database.seeds.user_seeder**Description****Classes**

```
UserSeeder([rows])
```

database.seeds.user_seeder.UserSeeder

```
class database.seeds.user_seeder.UserSeeder(rows: int = 30)
    Bases: object
```

Attributes

```
UserSeeder.name
```

database.seeds.user_seeder.UserSeeder.name

```
UserSeeder.name = 'UserSeeder'
```

Methods

```
UserSeeder.__init__([rows])
```

database.seeds.user_seeder.UserSeeder.__init__

UserSeeder.__init__(rows: int = 30)

class database.seeds.user_seeder.UserSeeder(rows: int = 30)

static _create_admin_user()

name = 'UserSeeder'

Functions

get_seeders()

init_seed()

database.seeds.get_seeders

database.seeds.get_seeders() → list

database.seeds.init_seed

database.seeds.init_seed() → None

database.seeds.get_seeders() → list

database.seeds.init_seed() → None

Functions

init_database()

seed_actions(fnc)

2.2.4 database.init_database

database.init_database() → None

2.2.5 database.seed_actions

database.seed_actions(*func*)

database.init_database() → None

database.seed_actions(*func*)

2.3 tests

Description

Package for testing the application.

The tests package stores the application's tests that are executed for ensuring the proper behaviour of the application.

You can get report coverage statistics with coverage package.

Notes

There are three kinds of tests created:

1. **Unit testing** Unit testing means testing individual modules of an application in isolation (without any interaction with dependencies) to confirm that the code is doing things right.
2. **Integration testing** Integration testing means checking if different modules are working fine when combined together as a group.
3. **Functional testing** Functional testing means testing a slice of functionality in the system (may interact with dependencies) to confirm that the code is doing the right things.

Let us understand these three types of testing with an oversimplified example.

E.g. For a functional mobile phone, the main parts required are “battery” and “sim card”.

Unit testing Example – The battery is checked for its life, capacity and other parameters. Sim card is checked for its activation.

Integration Testing Example – Battery and sim card are integrated i.e. assembled in order to start the mobile phone.

Functional Testing Example – The functionality of a mobile phone is checked in terms of its features and battery usage as well as sim card facilities.

References

[The Differences Between Unit Testing, Integration Testing and Functional Testing.](#)

Examples

How to usage:

```
source venv/bin/activate
pytest
```

How to call a specific test:

```
source venv/bin/activate
pytest tests/blueprints/test_base.py
```

How to call a specific test function:

```
source venv/bin/activate
pytest -k test_welcome_api
```

You can use coverage package for running tests as well:

```
source venv/bin/activate
coverage run -m pytest
```

And get a report coverage statistics on modules:

```
source venv/bin/activate
coverage report -m
```

For a nicer presentation, use coverage html to get annotated HTML listings detailing missed lines:

```
source venv/bin/activate
coverage html
```

References

How can I test Celery tasks?

Depends on what exactly you want to be testing.

- Test the task code directly. Don't call "task.delay(...)" just call "task(...)" from your unit tests.
- Use CELERY_ALWAYS_EAGER. This will cause your tasks to be called immediately at the point you say "task.delay(...)", so you can test the whole path (but not any asynchronous behavior).

<https://stackoverflow.com/questions/12078667/how-do-you-unit-test-a-celery-task>

Modules

<code>tests.blueprints</code>	Package for testing blueprints.
<code>tests.celery</code>	Package for testing Celery tasks.
<code>tests.conftest</code>	Module for configuring Pytest.
<code>tests.custom_flask_client</code>	
<code>tests.test_config</code>	Module for testing Config module.

continues on next page

Table 183 – continued from previous page

<code>tests.test_db</code>	Module for testing database.
<code>tests.test_mail</code>	Module for testing mail.
<code>tests.test_middleware</code>	

2.3.1 tests.blueprints

Description

Package for testing blueprints.

Modules

<code>tests.blueprints.test_auth</code>	Module for testing auth blueprint.
<code>tests.blueprints.test_base</code>	Module for testing base blueprint.
<code>tests.blueprints.test_documents</code>	Module for testing documents blueprint.
<code>tests.blueprints.test_roles</code>	Module for testing roles blueprint.
<code>tests.blueprints.test_tasks</code>	
<code>tests.blueprints.test_users</code>	Module for testing users blueprint.

tests.blueprints.test_auth

Description

Module for testing auth blueprint.

Functions

<code>test_request_reset_password(client)</code>
<code>test_reset_password(client)</code>
<code>test_user_login(client)</code>
<code>test_user_logout(client, auth_header)</code>
<code>test_validate_reset_password(client, app)</code>

tests.blueprints.test_auth.test_request_reset_password

```
tests.blueprints.test_auth.test_request_reset_password(client:
                                                    tests.custom_flask_client.CustomFlaskClient)
```

tests.blueprints.test_auth.test_reset_password

```
tests.blueprints.test_auth.test_reset_password(client: tests.custom_flask_client.CustomFlaskClient)
```

tests.blueprints.test_auth.test_user_login

```
tests.blueprints.test_auth.test_user_login(client: tests.custom_flask_client.CustomFlaskClient)
```

tests.blueprints.test_auth.test_user_logout

```
tests.blueprints.test_auth.test_user_logout(client: tests.custom_flask_client.CustomFlaskClient,
                                             auth_header: any)
```

tests.blueprints.test_auth.test_validate_reset_password

```
tests.blueprints.test_auth.test_validate_reset_password(client:
                                                       tests.custom_flask_client.CustomFlaskClient,
                                                       app: flask.app.Flask)
```

```
tests.blueprints.test_auth.test_request_reset_password(client:
                                                       tests.custom_flask_client.CustomFlaskClient)
```

```
tests.blueprints.test_auth.test_reset_password(client: tests.custom_flask_client.CustomFlaskClient)
```

```
tests.blueprints.test_auth.test_user_login(client: tests.custom_flask_client.CustomFlaskClient)
```

```
tests.blueprints.test_auth.test_user_logout(client: tests.custom_flask_client.CustomFlaskClient,
                                             auth_header: any)
```

```
tests.blueprints.test_auth.test_validate_reset_password(client:
                                                       tests.custom_flask_client.CustomFlaskClient,
                                                       app: flask.app.Flask)
```

tests.blueprints.test_base**Description**

Module for testing base blueprint.

Functions

```
test_welcome_api(client)
```

tests.blueprints.test_base.test_welcome_api

```
tests.blueprints.test_base.test_welcome_api(client: tests.custom_flask_client.CustomFlaskClient)
```

```
tests.blueprints.test_base.test_welcome_api(client: tests.custom_flask_client.CustomFlaskClient)
```

tests.blueprints.test_documents

Description

Module for testing documents blueprint.

Functions

```
test_delete_document(client, auth_header)
```

```
test_get_document_data(client, auth_header)
```

```
test_get_document_file(client, auth_header)
```

```
test_save_document(client, auth_header)
```

```
test_search_document(client, auth_header)
```

```
test_update_document(client, auth_header)
```

tests.blueprints.test_documents.test_delete_document

```
tests.blueprints.test_documents.test_delete_document(client:  
                                                    tests.custom_flask_client.CustomFlaskClient,  
                                                    auth_header: any)
```

tests.blueprints.test_documents.test_get_document_data

```
tests.blueprints.test_documents.test_get_document_data(client:
                                                         tests.custom_flask_client.CustomFlaskClient,
                                                         auth_header: any)
```

tests.blueprints.test_documents.test_get_document_file

```
tests.blueprints.test_documents.test_get_document_file(client:
                                                         tests.custom_flask_client.CustomFlaskClient,
                                                         auth_header: any)
```

tests.blueprints.test_documents.test_save_document

```
tests.blueprints.test_documents.test_save_document(client:
                                                     tests.custom_flask_client.CustomFlaskClient,
                                                     auth_header: any)
```

tests.blueprints.test_documents.test_search_document

```
tests.blueprints.test_documents.test_search_document(client:
                                                      tests.custom_flask_client.CustomFlaskClient,
                                                      auth_header: any)
```

tests.blueprints.test_documents.test_update_document

```
tests.blueprints.test_documents.test_update_document(client:
                                                      tests.custom_flask_client.CustomFlaskClient,
                                                      auth_header: any)
```

```
tests.blueprints.test_documents.test_delete_document(client:
                                                      tests.custom_flask_client.CustomFlaskClient,
                                                      auth_header: any)
```

```
tests.blueprints.test_documents.test_get_document_data(client:
                                                         tests.custom_flask_client.CustomFlaskClient,
                                                         auth_header: any)
```

```
tests.blueprints.test_documents.test_get_document_file(client:
                                                         tests.custom_flask_client.CustomFlaskClient,
                                                         auth_header: any)
```

```
tests.blueprints.test_documents.test_save_document(client:
                                                     tests.custom_flask_client.CustomFlaskClient,
                                                     auth_header: any)
```

```
tests.blueprints.test_documents.test_search_document(client:
                                                      tests.custom_flask_client.CustomFlaskClient,
                                                      auth_header: any)
```

```
tests.blueprints.test_documents.test_update_document(client:
                                                      tests.custom_flask_client.CustomFlaskClient,
                                                      auth_header: any)
```

tests.blueprints.test_roles

Description

Module for testing roles blueprint.

Functions

`test_delete_role_endpoint(client, auth_header)`

`test_get_role_endpoint(client, auth_header)`

`test_save_role_endpoint(client, auth_header, ...)`

`test_search_roles_endpoint(client, auth_header)`

`test_update_role_endpoint(client, ...)`

tests.blueprints.test_roles.test_delete_role_endpoint

`tests.blueprints.test_roles.test_delete_role_endpoint`(*client*:
tests.custom_flask_client.CustomFlaskClient,
auth_header: any)

tests.blueprints.test_roles.test_get_role_endpoint

`tests.blueprints.test_roles.test_get_role_endpoint`(*client*:
tests.custom_flask_client.CustomFlaskClient,
auth_header: any)

tests.blueprints.test_roles.test_save_role_endpoint

`tests.blueprints.test_roles.test_save_role_endpoint`(*client*:
tests.custom_flask_client.CustomFlaskClient,
auth_header: any, *factory*: any)

tests.blueprints.test_roles.test_search_roles_endpoint

`tests.blueprints.test_roles.test_search_roles_endpoint`(*client*:
tests.custom_flask_client.CustomFlaskClient,
auth_header: any)

tests.blueprints.test_roles.test_update_role_endpoint

```
tests.blueprints.test_roles.test_update_role_endpoint(client:
    tests.custom_flask_client.CustomFlaskClient,
    auth_header: any, factory: any)

tests.blueprints.test_roles.test_delete_role_endpoint(client:
    tests.custom_flask_client.CustomFlaskClient,
    auth_header: any)

tests.blueprints.test_roles.test_get_role_endpoint(client:
    tests.custom_flask_client.CustomFlaskClient,
    auth_header: any)

tests.blueprints.test_roles.test_save_role_endpoint(client:
    tests.custom_flask_client.CustomFlaskClient,
    auth_header: any, factory: any)

tests.blueprints.test_roles.test_search_roles_endpoint(client:
    tests.custom_flask_client.CustomFlaskClient,
    auth_header: any)

tests.blueprints.test_roles.test_update_role_endpoint(client:
    tests.custom_flask_client.CustomFlaskClient,
    auth_header: any, factory: any)
```

tests.blueprints.test_tasks**Description****Functions**

```
test_check_task_status(client, auth_header)
```

tests.blueprints.test_tasks.test_check_task_status

```
tests.blueprints.test_tasks.test_check_task_status(client:
    tests.custom_flask_client.CustomFlaskClient,
    auth_header: any)

tests.blueprints.test_tasks._create_task_record()

tests.blueprints.test_tasks.test_check_task_status(client:
    tests.custom_flask_client.CustomFlaskClient,
    auth_header: any)
```

tests.blueprints.test_users**Description**

Module for testing users blueprint.

Functions

`test_create_invalid_user_endpoint(client, ...)`

`test_create_user_endpoint(client, ...)`

`test_delete_user_endpoint(client, auth_header)`

`test_export_excel_and_word_endpoint(client,
...)`

`test_export_excel_endpoint(client, auth_header)`

`test_export_word_endpoint(client, auth_header)`

`test_get_user_endpoint(client, auth_header)`

`test_search_users_endpoint(client, auth_header)`

`test_update_user_endpoint(client, ...)`

tests.blueprints.test_users.test_create_invalid_user_endpoint

`tests.blueprints.test_users.test_create_invalid_user_endpoint` (*client*:
tests.custom_flask_client.CustomFlaskClient,
auth_header: any)

tests.blueprints.test_users.test_create_user_endpoint

`tests.blueprints.test_users.test_create_user_endpoint` (*client*:
tests.custom_flask_client.CustomFlaskClient,
auth_header: any, *factory*: any)

tests.blueprints.test_users.test_delete_user_endpoint

`tests.blueprints.test_users.test_delete_user_endpoint` (*client*:
tests.custom_flask_client.CustomFlaskClient,
auth_header: any)

tests.blueprints.test_users.test_export_excel_and_word_endpoint

```
tests.blueprints.test_users.test_export_excel_and_word_endpoint(client:
                                                                    tests.custom_flask_client.CustomFlaskClient,
                                                                    auth_header: any)
```

tests.blueprints.test_users.test_export_excel_endpoint

```
tests.blueprints.test_users.test_export_excel_endpoint(client:
                                                         tests.custom_flask_client.CustomFlaskClient,
                                                         auth_header: any)
```

tests.blueprints.test_users.test_export_word_endpoint

```
tests.blueprints.test_users.test_export_word_endpoint(client:
                                                         tests.custom_flask_client.CustomFlaskClient,
                                                         auth_header: any)
```

tests.blueprints.test_users.test_get_user_endpoint

```
tests.blueprints.test_users.test_get_user_endpoint(client:
                                                     tests.custom_flask_client.CustomFlaskClient,
                                                     auth_header: any)
```

tests.blueprints.test_users.test_search_users_endpoint

```
tests.blueprints.test_users.test_search_users_endpoint(client:
                                                         tests.custom_flask_client.CustomFlaskClient,
                                                         auth_header: any)
```

tests.blueprints.test_users.test_update_user_endpoint

```
tests.blueprints.test_users.test_update_user_endpoint(client:
                                                         tests.custom_flask_client.CustomFlaskClient,
                                                         auth_header: any, factory: any)
```

```
tests.blueprints.test_users.test_create_invalid_user_endpoint(client:
                                                                tests.custom_flask_client.CustomFlaskClient,
                                                                auth_header: any)
```

```
tests.blueprints.test_users.test_create_user_endpoint(client:
                                                         tests.custom_flask_client.CustomFlaskClient,
                                                         auth_header: any, factory: any)
```

```
tests.blueprints.test_users.test_delete_user_endpoint(client:
                                                         tests.custom_flask_client.CustomFlaskClient,
                                                         auth_header: any)
```

```
tests.blueprints.test_users.test_export_excel_and_word_endpoint(client:
                                                                    tests.custom_flask_client.CustomFlaskClient,
                                                                    auth_header: any)
```

```

tests.blueprints.test_users.test_export_excel_endpoint(client:
    tests.custom_flask_client.CustomFlaskClient,
    auth_header: any)

tests.blueprints.test_users.test_export_word_endpoint(client:
    tests.custom_flask_client.CustomFlaskClient,
    auth_header: any)

tests.blueprints.test_users.test_get_user_endpoint(client:
    tests.custom_flask_client.CustomFlaskClient,
    auth_header: any)

tests.blueprints.test_users.test_search_users_endpoint(client:
    tests.custom_flask_client.CustomFlaskClient,
    auth_header: any)

tests.blueprints.test_users.test_update_user_endpoint(client:
    tests.custom_flask_client.CustomFlaskClient,
    auth_header: any, factory: any)

```

2.3.2 tests.celery

Description

Package for testing Celery tasks.

Modules

<code>tests.celery.test_celery</code>	
<code>tests.celery.test_excel</code>	Module for testing excel module.
<code>tests.celery.test_tasks</code>	Module for testing task module.
<code>tests.celery.test_word</code>	Module for testing word module.

tests.celery.test_celery

Description

Functions

<code>test_register_context_task(app)</code>
--

tests.celery.test_celery.test_register_context_task

tests.celery.test_celery.test_register_context_task(app: flask.app.Flask)

tests.celery.test_celery.test_register_context_task(app: flask.app.Flask)

tests.celery.test_excel

Description

Module for testing excel module.

Functions

test_export_excel_task(app)

tests.celery.test_excel.test_export_excel_task

tests.celery.test_excel.test_export_excel_task(app: flask.app.Flask)

tests.celery.test_excel.test_export_excel_task(app: flask.app.Flask)

tests.celery.test_tasks

Description

Module for testing task module.

Functions

test_create_user_email_task(factory)

test_create_word_and_excel_documents(app)

test_reset_password_email_task(app)

test_send_email_with_attachments_task(app)

tests.celery.test_tasks.test_create_user_email_task

```
tests.celery.test_tasks.test_create_user_email_task(factory: any)
```

tests.celery.test_tasks.test_create_word_and_excel_documents

```
tests.celery.test_tasks.test_create_word_and_excel_documents(app: flask.app.Flask)
```

tests.celery.test_tasks.test_reset_password_email_task

```
tests.celery.test_tasks.test_reset_password_email_task(app: flask.app.Flask)
```

tests.celery.test_tasks.test_send_email_with_attachments_task

```
tests.celery.test_tasks.test_send_email_with_attachments_task(app: flask.app.Flask)
```

```
tests.celery.test_tasks.test_create_user_email_task(factory: any)
```

```
tests.celery.test_tasks.test_create_word_and_excel_documents(app: flask.app.Flask)
```

```
tests.celery.test_tasks.test_reset_password_email_task(app: flask.app.Flask)
```

```
tests.celery.test_tasks.test_send_email_with_attachments_task(app: flask.app.Flask)
```

tests.celery.test_word**Description**

Module for testing word module.

Functions

```
test_export_word_task(app)
```

tests.celery.test_word.test_export_word_task

```
tests.celery.test_word.test_export_word_task(app: flask.app.Flask)
```

```
tests.celery.test_word.test_export_word_task(app: flask.app.Flask)
```

2.3.3 tests.conftest

Description

Module for configuring Pytest.

Functions

<code>app()</code>	Create an app with testing environment.
<code>auth_header(app, client)</code>	Create an auth header from a given user that can be added to an http requests.
<code>client(app)</code>	Create a test client for making http requests.
<code>factory(app)</code>	Create a Factory from a database model.
<code>runner(app)</code>	Create a CLI runner for testing CLI commands.

tests.conftest.app

`tests.conftest.app()`
Create an app with testing environment.

tests.conftest.auth_header

`tests.conftest.auth_header(app: flask.app.Flask, client: tests.custom_flask_client.CustomFlaskClient)`
Create an auth header from a given user that can be added to an http requests.

tests.conftest.client

`tests.conftest.client(app: flask.app.Flask)`
Create a test client for making http requests.

tests.conftest.factory

`tests.conftest.factory(app: flask.app.Flask)`
Create a Factory from a database model.

tests.conftest.runner

`tests.conftest.runner(app: flask.app.Flask)`
Create a CLI runner for testing CLI commands.

`tests.conftest._remove_test_files(storage_path: str) → None`
Remove test files created in storage path.

`tests.conftest.app()`
Create an app with testing environment.

`tests.conftest.auth_header(app: flask.app.Flask, client: tests.custom_flask_client.CustomFlaskClient)`
Create an auth header from a given user that can be added to an http requests.

`tests.conftest.client(app: flask.app.Flask)`
Create a test client for making http requests.

`tests.conftest.factory(app: flask.app.Flask)`
Create a Factory from a database model.

`tests.conftest.runner(app: flask.app.Flask)`
Create a CLI runner for testing CLI commands.

2.3.4 tests.custom_flask_client

Description

Classes

*CustomFlaskClient(*args, **kwargs)*

tests.custom_flask_client.CustomFlaskClient

`class tests.custom_flask_client.CustomFlaskClient(*args: Any, **kwargs: Any)`
Bases: flask.testing.FlaskClient

Attributes

CustomFlaskClient.preserve_context

tests.custom_flask_client.CustomFlaskClient.preserve_context

`CustomFlaskClient.preserve_context = False`

Methods

<i>CustomFlaskClient.__init__(*args, **kwargs)</i>	
<i>CustomFlaskClient.after_request(response)</i>	
<i>CustomFlaskClient.before_request(*args, **kwargs)</i>	
<i>CustomFlaskClient.delete(*args, **kwargs)</i>	Like open but method is enforced to DELETE.
<i>CustomFlaskClient.delete_cookie(server_name, key)</i>	Deletes a cookie in the test client.
<i>CustomFlaskClient.get(*args, **kwargs)</i>	Like open but method is enforced to GET.
<i>CustomFlaskClient.head(*args, **kw)</i>	Call <i>open()</i> with method set to HEAD.

continues on next page

Table 199 – continued from previous page

<code>CustomFlaskClient. make_request(method, ...)</code>	
<code>CustomFlaskClient.open(*args[, as_tuple, ...])</code>	Generate an environ dict from the given arguments, make a request to the application using it, and return the response.
<code>CustomFlaskClient.options(*args, **kw)</code>	Call <code>open()</code> with method set to OPTIONS.
<code>CustomFlaskClient.patch(*args, **kw)</code>	Call <code>open()</code> with method set to PATCH.
<code>CustomFlaskClient.post(*args, **kwargs)</code>	Like open but method is enforced to POST.
<code>CustomFlaskClient.put(*args, **kwargs)</code>	Like open but method is enforced to PUT.
<code>CustomFlaskClient. resolve_redirect(response)</code>	Perform a new request to the location given by the redirect response to the previous request.
<code>CustomFlaskClient. run_wsgi_app(environ[, ...])</code>	Runs the wrapped WSGI app with the given environment.
<code>CustomFlaskClient. session_transaction(*args, ...)</code>	When used in combination with a <code>with</code> statement this opens a session transaction.
<code>CustomFlaskClient. set_cookie(server_name, key)</code>	Sets a cookie in the client's cookie jar.
<code>CustomFlaskClient.trace(*args, **kw)</code>	Call <code>open()</code> with method set to TRACE.

tests.custom_flask_client.CustomFlaskClient.__init__

`CustomFlaskClient.__init__(*args: Any, **kwargs: Any) → None`

tests.custom_flask_client.CustomFlaskClient.after_request

static `CustomFlaskClient.after_request(response: flask.wrappers.Response)`

tests.custom_flask_client.CustomFlaskClient.before_request

static `CustomFlaskClient.before_request(*args, **kwargs)`

tests.custom_flask_client.CustomFlaskClient.delete

`CustomFlaskClient.delete(*args, **kwargs)`
Like open but method is enforced to DELETE.

tests.custom_flask_client.CustomFlaskClient.delete_cookie

`CustomFlaskClient.delete_cookie(server_name: str, key: str, path: str = '/', domain:
Optional[str] = None, secure: bool = False, httponly:
bool = False, samesite: Optional[str] = None) → None`
Deletes a cookie in the test client.

tests.custom_flask_client.CustomFlaskClient.get

`CustomFlaskClient.get(*args, **kwargs)`

Like `open` but method is enforced to GET.

tests.custom_flask_client.CustomFlaskClient.head

`CustomFlaskClient.head(*args: Any, **kw: Any) → werkzeug.test.TestResponse`

Call `open()` with method set to HEAD.

tests.custom_flask_client.CustomFlaskClient.make_request

`CustomFlaskClient.make_request(method: str, *args, **kwargs)`

tests.custom_flask_client.CustomFlaskClient.open

`CustomFlaskClient.open(*args: Any, as_tuple: bool = False, buffered: bool = False, follow_redirects: bool = False, **kwargs: Any) → Response`

Generate an environ dict from the given arguments, make a request to the application using it, and return the response.

Parameters

- **args** – Passed to `EnvironBuilder` to create the environ for the request. If a single arg is passed, it can be an existing `EnvironBuilder` or an environ dict.
- **buffered** – Convert the iterator returned by the app into a list. If the iterator has a `close()` method, it is called automatically.
- **follow_redirects** – Make additional requests to follow HTTP redirects until a non-redirect status is returned. `TestResponse.history` lists the intermediate responses.

Changed in version 2.0: `as_tuple` is deprecated and will be removed in Werkzeug 2.1. Use `TestResponse.request` and `request.environ` instead.

Changed in version 2.0: The request input stream is closed when calling `response.close()`. Input streams for redirects are automatically closed.

Changed in version 0.5: If a dict is provided as file in the dict for the data parameter the content type has to be called `content_type` instead of `mimetype`. This change was made for consistency with `werkzeug.FileWrapper`.

Changed in version 0.5: Added the `follow_redirects` parameter.

tests.custom_flask_client.CustomFlaskClient.options

`CustomFlaskClient.options(*args: Any, **kw: Any) → werkzeug.test.TestResponse`

Call `open()` with method set to OPTIONS.

tests.custom_flask_client.CustomFlaskClient.patch

`CustomFlaskClient.patch(*args: Any, **kw: Any) → werkzeug.test.TestResponse`
Call `open()` with method set to PATCH.

tests.custom_flask_client.CustomFlaskClient.post

`CustomFlaskClient.post(*args, **kwargs)`
Like open but method is enforced to POST.

tests.custom_flask_client.CustomFlaskClient.put

`CustomFlaskClient.put(*args, **kwargs)`
Like open but method is enforced to PUT.

tests.custom_flask_client.CustomFlaskClient.resolve_redirect

`CustomFlaskClient.resolve_redirect(response: werkzeug.test.TestResponse, buffered: bool = False) → werkzeug.test.TestResponse`
Perform a new request to the location given by the redirect response to the previous request.

tests.custom_flask_client.CustomFlaskClient.run_wsgi_app

`CustomFlaskClient.run_wsgi_app(environ: WSGIEnvironment, buffered: bool = False) → Tuple[Iterable[bytes], str, werkzeug.datastructures.Headers]`
Runs the wrapped WSGI app with the given environment.

tests.custom_flask_client.CustomFlaskClient.session_transaction

`CustomFlaskClient.session_transaction(*args: Any, **kwargs: Any) → Generator[flask.sessions.SessionMixin, None, None]`

When used in combination with a `with` statement this opens a session transaction. This can be used to modify the session that the test client uses. Once the `with` block is left the session is stored back.

```
with client.session_transaction() as session:
    session['value'] = 42
```

Internally this is implemented by going through a temporary test request context and since session handling could depend on request variables this function accepts the same arguments as `test_request_context()` which are directly passed through.

tests.custom_flask_client.CustomFlaskClient.set_cookie

`CustomFlaskClient.set_cookie(server_name: str, key: str, value: str = "", max_age: Optional[Union[datetime.timedelta, int]] = None, expires: Optional[Union[str, datetime.datetime, int, float]] = None, path: str = '/', domain: Optional[str] = None, secure: bool = False, httponly: bool = False, samesite: Optional[str] = None, charset: str = 'utf-8') → None`

Sets a cookie in the client's cookie jar. The server name is required and has to match the one that is also passed to the open call.

tests.custom_flask_client.CustomFlaskClient.trace

`CustomFlaskClient.trace(*args: Any, **kw: Any) → werkzeug.test.TestResponse`
Call `open()` with method set to TRACE.

class `tests.custom_flask_client.CustomFlaskClient(*args: Any, **kwargs: Any)`

static `after_request(response: flask.wrappers.Response)`

application: `Flask`

static `before_request(*args, **kwargs)`

delete(`*args, **kwargs`)

Like open but method is enforced to DELETE.

delete_cookie(`server_name: str, key: str, path: str = '/', domain: Optional[str] = None, secure: bool = False, httponly: bool = False, samesite: Optional[str] = None`) → None

Deletes a cookie in the test client.

get(`*args, **kwargs`)

Like open but method is enforced to GET.

head(`*args: Any, **kw: Any`) → `werkzeug.test.TestResponse`

Call `open()` with method set to HEAD.

make_request(`method: str, *args, **kwargs`)

open(`*args: Any, as_tuple: bool = False, buffered: bool = False, follow_redirects: bool = False, **kwargs: Any`) → `Response`

Generate an environ dict from the given arguments, make a request to the application using it, and return the response.

Parameters

- **args** – Passed to `EnvironBuilder` to create the environ for the request. If a single arg is passed, it can be an existing `EnvironBuilder` or an environ dict.
- **buffered** – Convert the iterator returned by the app into a list. If the iterator has a `close()` method, it is called automatically.
- **follow_redirects** – Make additional requests to follow HTTP redirects until a non-redirect status is returned. `TestResponse.history` lists the intermediate responses.

Changed in version 2.0: `as_tuple` is deprecated and will be removed in Werkzeug 2.1. Use `TestResponse.request` and `request.environ` instead.

Changed in version 2.0: The request input stream is closed when calling `response.close()`. Input streams for redirects are automatically closed.

Changed in version 0.5: If a dict is provided as file in the dict for the `data` parameter the content type has to be called `content_type` instead of `mimetype`. This change was made for consistency with `werkzeug.FileWrapper`.

Changed in version 0.5: Added the `follow_redirects` parameter.

options(*args: Any, **kw: Any) → `werkzeug.test.TestResponse`
Call `open()` with method set to `OPTIONS`.

patch(*args: Any, **kw: Any) → `werkzeug.test.TestResponse`
Call `open()` with method set to `PATCH`.

post(*args, **kwargs)
Like open but method is enforced to `POST`.

preserve_context = `False`

put(*args, **kwargs)
Like open but method is enforced to `PUT`.

resolve_redirect(response: `werkzeug.test.TestResponse`, buffered: `bool = False`) → `werkzeug.test.TestResponse`
Perform a new request to the location given by the redirect response to the previous request.

run_wsgi_app(environ: `WSGIEnvironment`, buffered: `bool = False`) → `Tuple[Iterable[bytes], str, werkzeug.datastructures.Headers]`
Runs the wrapped WSGI app with the given environment.

session_transaction(*args: Any, **kwargs: Any) → `Generator[flask.sessions.SessionMixin, None, None]`

When used in combination with a `with` statement this opens a session transaction. This can be used to modify the session that the test client uses. Once the `with` block is left the session is stored back.

```
with client.session_transaction() as session:
    session['value'] = 42
```

Internally this is implemented by going through a temporary test request context and since session handling could depend on request variables this function accepts the same arguments as `test_request_context()` which are directly passed through.

set_cookie(server_name: `str`, key: `str`, value: `str` = "", max_age: `Optional[Union[datetime.timedelta, int]]` = `None`, expires: `Optional[Union[str, datetime.datetime, int, float]]` = `None`, path: `str` = '/', domain: `Optional[str]` = `None`, secure: `bool` = `False`, httponly: `bool` = `False`, samesite: `Optional[str]` = `None`, charset: `str` = 'utf-8') → `None`

Sets a cookie in the client's cookie jar. The server name is required and has to match the one that is also passed to the open call.

trace(*args: Any, **kw: Any) → `werkzeug.test.TestResponse`
Call `open()` with method set to `TRACE`.

2.3.5 tests.test_config

Description

Module for testing Config module.

Functions

<code>test_config()</code>	Check if TESTING attribute is enabled.
----------------------------	--

tests.test_config.test_config

`tests.test_config.test_config()`
Check if TESTING attribute is enabled.

`tests.test_config.test_config()`
Check if TESTING attribute is enabled.

2.3.6 tests.test_db

Description

Module for testing database.

Functions

<code>test_get_close_db()</code>	Check if a database connection is closed.
----------------------------------	---

tests.test_db.test_get_close_db

`tests.test_db.test_get_close_db()`
Check if a database connection is closed.

`tests.test_db.test_get_close_db()`
Check if a database connection is closed.

2.3.7 tests.test_mail

Description

Module for testing mail.

Functions

<code>test_mail_record_messages(app)</code>	Check if a email is sent.
---	---------------------------

tests.test_mail.test_mail_record_messages

`tests.test_mail.test_mail_record_messages(app)`
Check if a email is sent.

References

Unit tests and suppressing emails

`tests.test_mail.test_mail_record_messages(app)`
Check if a email is sent.

References

Unit tests and suppressing emails

2.3.8 tests.test_middleware

Description

Functions

<code>test_api_middleware(client, auth_header)</code>

<code>test_no_api_middleware(client)</code>

tests.test_middleware.test_api_middleware

`tests.test_middleware.test_api_middleware(client: tests.custom_flask_client.CustomFlaskClient, auth_header: any)`

tests.test_middleware.test_no_api_middleware

`tests.test_middleware.test_no_api_middleware(client: tests.custom_flask_client.CustomFlaskClient)`

`tests.test_middleware.test_api_middleware(client: tests.custom_flask_client.CustomFlaskClient, auth_header: any)`

`tests.test_middleware.test_no_api_middleware(client: tests.custom_flask_client.CustomFlaskClient)`

2.4 config

Description

Module loads the application's configuration.

The extension and custom configurations are defined here.

Classes

<i>Config()</i>	Default configuration options.
<i>DevConfig()</i>	Development configuration options.
<i>Meta</i> (name, bases, dict)	Metaclass for updating Config options.
<i>ProdConfig()</i>	Production configuration options.
<i>TestConfig()</i>	Testing configuration options.

2.4.1 config.Config

class config.Config

Bases: object

Default configuration options.

Attributes

Config.ALLOWED_CONTENT_TYPES

Config.ALLOWED_MIME_TYPES

Config.DATABASE

Config.DEBUG

Config.DEVELOPMENT

Config.FLASK_RESTFUL_PREFIX

Config.HOME

Config.LOGIN_DISABLED

Config.LOG_DIRECTORY

Config.MAIL_PASSWORD

Config.MAIL_PORT

continues on next page

Table 205 – continued from previous page

<i>Config.MAIL_SERVER</i>
<i>Config.MAIL_USERNAME</i>
<i>Config.MAIL_USE_SSL</i>
<i>Config.MAIL_USE_TLS</i>
<i>Config.MOCKUP_DIRECTORY</i>
<i>Config.RESET_TOKEN_EXPIRES</i>
<i>Config.RESTX_ERROR_404_HELP</i>
<i>Config.RESTX_MASK_SWAGGER</i>
<i>Config.ROOT_DIRECTORY</i>
<i>Config.SECRET_KEY</i>
<i>Config.SECURITY_PASSWORD_HASH</i>
<i>Config.SECURITY_PASSWORD_LENGTH_MIN</i>
<i>Config.SECURITY_PASSWORD_SALT</i>
<i>Config.SECURITY_TOKEN_AUTHENTICATION_HEADER</i>
<i>Config.SECURITY_TOKEN_MAX_AGE</i>
<i>Config.SERVER_NAME</i>
<i>Config.STORAGE_DIRECTORY</i>
<i>Config.SWAGGER_API_URL</i>
<i>Config.SWAGGER_URL</i>
<i>Config.TESTING</i>
<i>Config.TEST_USER_EMAIL</i>
<i>Config.TEST_USER_PASSWORD</i>
<i>Config.accept_content</i>
<i>Config.broker_url</i>
<i>Config.enable_utc</i>

continues on next page

Table 205 – continued from previous page

<i>Config.include</i>
<i>Config.result_backend</i>
<i>Config.result_expires</i>
<i>Config.result_extended</i>
<i>Config.result_serializer</i>
<i>Config.task_always_eager</i>
<i>Config.task_default_rate_limit</i>
<i>Config.task_serializer</i>
<i>Config.task_track_started</i>
<i>Config.timezone</i>
<i>Config.worker_log_format</i>
<i>Config.worker_task_log_format</i>

config.Config.ALLOWED_CONTENT_TYPES

```
Config.ALLOWED_CONTENT_TYPES = {'application/json',
                                'application/octet-stream', 'multipart/form-data'}
```

config.Config.ALLOWED_MIME_TYPES

```
Config.ALLOWED_MIME_TYPES = {'application/pdf', 'application/vnd.ms-excel'}
```

config.Config.DATABASE

```
Config.DATABASE = {'engine': 'peewee.SqliteDatabase', 'name': None,  
'pragmas': {'cache_size': -64000, 'foreign_keys': 1,  
'ignore_check_constraints': 0, 'journal_mode': 'wal', 'synchronous': 0}}
```

config.Config.DEBUG

```
Config.DEBUG = False
```

config.Config.DEVELOPMENT

```
Config.DEVELOPMENT = False
```

config.Config.FLASK_RESTFUL_PREFIX

```
Config.FLASK_RESTFUL_PREFIX = '/api'
```

config.Config.HOME

```
Config.HOME = '/home/docs'
```

config.Config.LOGIN_DISABLED

```
Config.LOGIN_DISABLED = False
```

config.Config.LOG_DIRECTORY

```
Config.LOG_DIRECTORY = '/home/docs/checkouts/readthedocs.org/user_builds/  
flask-api/checkouts/latest/log'
```

config.Config.MAIL_PASSWORD

```
Config.MAIL_PASSWORD = None
```

config.Config.MAIL_PORT

```
Config.MAIL_PORT = None
```

config.Config.MAIL_SERVER

Config.MAIL_SERVER = None

config.Config.MAIL_USERNAME

Config.MAIL_USERNAME = None

config.Config.MAIL_USE_SSL

Config.MAIL_USE_SSL = False

config.Config.MAIL_USE_TLS

Config.MAIL_USE_TLS = True

config.Config MOCKUP_DIRECTORY

Config.MOCKUP_DIRECTORY = '/home/docs/checkouts/readthedocs.org/user_builds/flask-api/checkouts/latest/storage/mockups'

config.Config.RESET_TOKEN_EXPIRES

Config.RESET_TOKEN_EXPIRES = 86400

config.Config.RESTX_ERROR_404_HELP

Config.RESTX_ERROR_404_HELP = False

config.Config.RESTX_MASK_SWAGGER

Config.RESTX_MASK_SWAGGER = False

config.Config.ROOT_DIRECTORY

Config.ROOT_DIRECTORY = '/home/docs/checkouts/readthedocs.org/user_builds/flask-api/checkouts/latest'

config.Config.SECRET_KEY

Config.SECRET_KEY = None

config.Config.SECURITY_PASSWORD_HASH

Config.SECURITY_PASSWORD_HASH = 'pbkdf2_sha512'

config.Config.SECURITY_PASSWORD_LENGTH_MIN

Config.SECURITY_PASSWORD_LENGTH_MIN = 8

config.Config.SECURITY_PASSWORD_SALT

Config.SECURITY_PASSWORD_SALT = None

config.Config.SECURITY_TOKEN_AUTHENTICATION_HEADER

Config.SECURITY_TOKEN_AUTHENTICATION_HEADER = 'Authorization'

config.Config.SECURITY_TOKEN_MAX_AGE

Config.SECURITY_TOKEN_MAX_AGE = None

config.Config.SERVER_NAME

Config.SERVER_NAME = None

config.Config.STORAGE_DIRECTORY

Config.STORAGE_DIRECTORY = '/home/docs/checkouts/readthedocs.org/
user_builds/flask-api/checkouts/latest/storage'

config.Config.SWAGGER_API_URL

Config.SWAGGER_API_URL = 'http://None/static/swagger.yaml'

config.Config.SWAGGER_URL

Config.SWAGGER_URL = '/docs'

config.Config.TESTING

```
Config.TESTING = False
```

config.Config.TEST_USER_EMAIL

```
Config.TEST_USER_EMAIL = None
```

config.Config.TEST_USER_PASSWORD

```
Config.TEST_USER_PASSWORD = None
```

config.Config.accept_content

```
Config.accept_content = ['json']
```

config.Config.broker_url

```
Config.broker_url = 'pyamqp://'
```

config.Config.enable_utc

```
Config.enable_utc = True
```

config.Config.include

```
Config.include = ['app.celery.tasks']
```

config.Config.result_backend

```
Config.result_backend = 'amqp://'
```

config.Config.result_expires

```
Config.result_expires = 3600
```

config.Config.result_extended

```
Config.result_extended = True
```

config.Config.result_serializer

```
Config.result_serializer = 'json'
```

config.Config.task_always_eager

```
Config.task_always_eager = False
```

config.Config.task_default_rate_limit

```
Config.task_default_rate_limit = 3
```

config.Config.task_serializer

```
Config.task_serializer = 'json'
```

config.Config.task_track_started

```
Config.task_track_started = True
```

config.Config.timezone

```
Config.timezone = 'UTC'
```

config.Config.worker_log_format

```
Config.worker_log_format = '%(asctime)s - %(levelname)s - %(processName)s -  
%(message)s'
```

config.Config.worker_task_log_format

```
Config.worker_task_log_format = '%(asctime)s - %(levelname)s -  
%(processName)s - %(task_name)s - %(task_id)s - %(message)s'
```

Methods

```
Config.__init__()
```

config.Config.__init__

Config.__init__()

2.4.2 config.DevConfig

class config.DevConfig

Bases: *config.Config*

Development configuration options.

Attributes

DevConfig.ALLOWED_CONTENT_TYPES

DevConfig.ALLOWED_MIME_TYPES

DevConfig.DATABASE

DevConfig.DEBUG

DevConfig.DEVELOPMENT

DevConfig.FLASK_RESTFUL_PREFIX

DevConfig.HOME

DevConfig.LOGIN_DISABLED

DevConfig.LOG_DIRECTORY

DevConfig.MAIL_PASSWORD

DevConfig.MAIL_PORT

DevConfig.MAIL_SERVER

DevConfig.MAIL_USERNAME

DevConfig.MAIL_USE_SSL

DevConfig.MAIL_USE_TLS

DevConfig.MOCKUP_DIRECTORY

DevConfig.RESET_TOKEN_EXPIRES

DevConfig.RESTX_ERROR_404_HELP

continues on next page

Table 207 – continued from previous page

<i>DevConfig.RESTX_MASK_SWAGGER</i>
<i>DevConfig.ROOT_DIRECTORY</i>
<i>DevConfig.SECRET_KEY</i>
<i>DevConfig.SECURITY_PASSWORD_HASH</i>
<i>DevConfig.SECURITY_PASSWORD_LENGTH_MIN</i>
<i>DevConfig.SECURITY_PASSWORD_SALT</i>
<i>DevConfig.SECURITY_TOKEN_AUTHENTICATION_HEADER</i>
<i>DevConfig.SECURITY_TOKEN_MAX_AGE</i>
<i>DevConfig.SERVER_NAME</i>
<i>DevConfig.STORAGE_DIRECTORY</i>
<i>DevConfig.SWAGGER_API_URL</i>
<i>DevConfig.SWAGGER_URL</i>
<i>DevConfig.TESTING</i>
<i>DevConfig.TEST_USER_EMAIL</i>
<i>DevConfig.TEST_USER_PASSWORD</i>
<i>DevConfig.accept_content</i>
<i>DevConfig.broker_url</i>
<i>DevConfig.enable_utc</i>
<i>DevConfig.include</i>
<i>DevConfig.result_backend</i>
<i>DevConfig.result_expires</i>
<i>DevConfig.result_extended</i>
<i>DevConfig.result_serializer</i>
<i>DevConfig.task_always_eager</i>
<i>DevConfig.task_default_rate_limit</i>

continues on next page

Table 207 – continued from previous page

DevConfig.task_serializer

DevConfig.task_track_started

DevConfig.timezone

DevConfig.worker_log_format

DevConfig.worker_task_log_format

config.DevConfig.ALLOWED_CONTENT_TYPES

```
DevConfig.ALLOWED_CONTENT_TYPES = {'application/json',
                                     'application/octet-stream', 'multipart/form-data'}
```

config.DevConfig.ALLOWED_MIME_TYPES

```
DevConfig.ALLOWED_MIME_TYPES = {'application/pdf',
                                  'application/vnd.ms-excel'}
```

config.DevConfig.DATABASE

```
DevConfig.DATABASE = {'engine': 'peewee.SqliteDatabase', 'name': None,
                       'pragmas': {'cache_size': -64000, 'foreign_keys': 1,
                                     'ignore_check_constraints': 0, 'journal_mode': 'wal', 'synchronous': 0}}
```

config.DevConfig.DEBUG

```
DevConfig.DEBUG = True
```

config.DevConfig.DEVELOPMENT

```
DevConfig.DEVELOPMENT = True
```

config.DevConfig.FLASK_RESTFUL_PREFIX

```
DevConfig.FLASK_RESTFUL_PREFIX = '/api'
```

config.DevConfig.HOME

```
DevConfig.HOME = '/home/docs'
```

config.DevConfig.LOGIN_DISABLED

```
DevConfig.LOGIN_DISABLED = False
```

config.DevConfig.LOG_DIRECTORY

```
DevConfig.LOG_DIRECTORY = '/home/docs/checkouts/readthedocs.org/user_builds/  
flask-api/checkouts/latest/log'
```

config.DevConfig.MAIL_PASSWORD

```
DevConfig.MAIL_PASSWORD = None
```

config.DevConfig.MAIL_PORT

```
DevConfig.MAIL_PORT = None
```

config.DevConfig.MAIL_SERVER

```
DevConfig.MAIL_SERVER = None
```

config.DevConfig.MAIL_USERNAME

```
DevConfig.MAIL_USERNAME = None
```

config.DevConfig.MAIL_USE_SSL

```
DevConfig.MAIL_USE_SSL = False
```

config.DevConfig.MAIL_USE_TLS

```
DevConfig.MAIL_USE_TLS = True
```

config.DevConfig MOCKUP_DIRECTORY

```
DevConfig.MOCKUP_DIRECTORY = '/home/docs/checkouts/readthedocs.org/  
user_builds/flask-api/checkouts/latest/storage/mockups'
```

config.DevConfig.RESET_TOKEN_EXPIRES

DevConfig.RESET_TOKEN_EXPIRES = 86400

config.DevConfig.RESTX_ERROR_404_HELP

DevConfig.RESTX_ERROR_404_HELP = False

config.DevConfig.RESTX_MASK_SWAGGER

DevConfig.RESTX_MASK_SWAGGER = False

config.DevConfig.ROOT_DIRECTORY

DevConfig.ROOT_DIRECTORY = '/home/docs/checkouts/readthedocs.org/
user_builds/flask-api/checkouts/latest'

config.DevConfig.SECRET_KEY

DevConfig.SECRET_KEY = None

config.DevConfig.SECURITY_PASSWORD_HASH

DevConfig.SECURITY_PASSWORD_HASH = 'pbkdf2_sha512'

config.DevConfig.SECURITY_PASSWORD_LENGTH_MIN

DevConfig.SECURITY_PASSWORD_LENGTH_MIN = 8

config.DevConfig.SECURITY_PASSWORD_SALT

DevConfig.SECURITY_PASSWORD_SALT = None

config.DevConfig.SECURITY_TOKEN_AUTHENTICATION_HEADER

DevConfig.SECURITY_TOKEN_AUTHENTICATION_HEADER = 'Authorization'

config.DevConfig.SECURITY_TOKEN_MAX_AGE

DevConfig.SECURITY_TOKEN_MAX_AGE = None

config.DevConfig.SERVER_NAME

```
DevConfig.SERVER_NAME = None
```

config.DevConfig.STORAGE_DIRECTORY

```
DevConfig.STORAGE_DIRECTORY = '/home/docs/checkouts/readthedocs.org/  
user_builds/flask-api/checkouts/latest/storage'
```

config.DevConfig.SWAGGER_API_URL

```
DevConfig.SWAGGER_API_URL = 'http://None/static/swagger.yaml'
```

config.DevConfig.SWAGGER_URL

```
DevConfig.SWAGGER_URL = '/docs'
```

config.DevConfig.TESTING

```
DevConfig.TESTING = False
```

config.DevConfig.TEST_USER_EMAIL

```
DevConfig.TEST_USER_EMAIL = None
```

config.DevConfig.TEST_USER_PASSWORD

```
DevConfig.TEST_USER_PASSWORD = None
```

config.DevConfig.accept_content

```
DevConfig.accept_content = ['json']
```

config.DevConfig.broker_url

```
DevConfig.broker_url = 'pyamqp://'
```

config.DevConfig.enable_utc

```
DevConfig.enable_utc = True
```


config.DevConfig.include

```
DevConfig.include = ['app.celery.tasks']
```

config.DevConfig.result_backend

```
DevConfig.result_backend = 'amqp://'
```

config.DevConfig.result_expires

```
DevConfig.result_expires = 3600
```

config.DevConfig.result_extended

```
DevConfig.result_extended = True
```

config.DevConfig.result_serializer

```
DevConfig.result_serializer = 'json'
```

config.DevConfig.task_always_eager

```
DevConfig.task_always_eager = False
```

config.DevConfig.task_default_rate_limit

```
DevConfig.task_default_rate_limit = 3
```

config.DevConfig.task_serializer

```
DevConfig.task_serializer = 'json'
```

config.DevConfig.task_track_started

```
DevConfig.task_track_started = True
```

config.DevConfig.timezone

```
DevConfig.timezone = 'UTC'
```

config.DevConfig.worker_log_format

```
DevConfig.worker_log_format = '%(asctime)s - %(levelname)s - %(processName)s  
- %(message)s'
```

config.DevConfig.worker_task_log_format

```
DevConfig.worker_task_log_format = '%(asctime)s - %(levelname)s -  
%(processName)s - %(task_name)s - %(task_id)s - %(message)s'
```

Methods

```
DevConfig.__init__()
```

config.DevConfig.__init__

```
DevConfig.__init__()
```

2.4.3 config.Meta

```
class config.Meta(name: str, bases: tuple, dict: dict)
```

Bases: type

Metaclass for updating Config options.

Methods

```
Meta.__init__(*args, **kwargs)
```

```
Meta.mro()
```

Return a type's method resolution order.

config.Meta.__init__

```
Meta.__init__(*args, **kwargs)
```

config.Meta.mro**Meta.mro()**

Return a type's method resolution order.

2.4.4 config.ProdConfig**class config.ProdConfig**Bases: *config.Config*

Production configuration options.

Attributes*ProdConfig.ALLOWED_CONTENT_TYPES**ProdConfig.ALLOWED_MIME_TYPES**ProdConfig.DATABASE**ProdConfig.DEBUG**ProdConfig.DEVELOPMENT**ProdConfig.FLASK_RESTFUL_PREFIX**ProdConfig.HOME**ProdConfig.LOGIN_DISABLED**ProdConfig.LOG_DIRECTORY**ProdConfig.MAIL_PASSWORD**ProdConfig.MAIL_PORT**ProdConfig.MAIL_SERVER**ProdConfig.MAIL_USERNAME**ProdConfig.MAIL_USE_SSL**ProdConfig.MAIL_USE_TLS**ProdConfig.MOCKUP_DIRECTORY**ProdConfig.RESET_TOKEN_EXPIRES**ProdConfig.RESTX_ERROR_404_HELP*

continues on next page

Table 210 – continued from previous page

<i>ProdConfig.RESTX_MASK_SWAGGER</i>
<i>ProdConfig.ROOT_DIRECTORY</i>
<i>ProdConfig.SECRET_KEY</i>
<i>ProdConfig.SECURITY_PASSWORD_HASH</i>
<i>ProdConfig.SECURITY_PASSWORD_LENGTH_MIN</i>
<i>ProdConfig.SECURITY_PASSWORD_SALT</i>
<i>ProdConfig.SECURITY_TOKEN_AUTHENTICATION_HEADER</i>
<i>ProdConfig.SECURITY_TOKEN_MAX_AGE</i>
<i>ProdConfig.SERVER_NAME</i>
<i>ProdConfig.STORAGE_DIRECTORY</i>
<i>ProdConfig.SWAGGER_API_URL</i>
<i>ProdConfig.SWAGGER_URL</i>
<i>ProdConfig.TESTING</i>
<i>ProdConfig.TEST_USER_EMAIL</i>
<i>ProdConfig.TEST_USER_PASSWORD</i>
<i>ProdConfig.accept_content</i>
<i>ProdConfig.broker_url</i>
<i>ProdConfig.enable_utc</i>
<i>ProdConfig.include</i>
<i>ProdConfig.result_backend</i>
<i>ProdConfig.result_expires</i>
<i>ProdConfig.result_extended</i>
<i>ProdConfig.result_serializer</i>
<i>ProdConfig.task_always_eager</i>
<i>ProdConfig.task_default_rate_limit</i>

continues on next page

Table 210 – continued from previous page

<i>ProdConfig.task_serializer</i>
<i>ProdConfig.task_track_started</i>
<i>ProdConfig.timezone</i>
<i>ProdConfig.worker_log_format</i>
<i>ProdConfig.worker_task_log_format</i>

config.ProdConfig.ALLOWED_CONTENT_TYPES

```
ProdConfig.ALLOWED_CONTENT_TYPES = {'application/json',
    'application/octet-stream', 'multipart/form-data'}
```

config.ProdConfig.ALLOWED_MIME_TYPES

```
ProdConfig.ALLOWED_MIME_TYPES = {'application/pdf',
    'application/vnd.ms-excel'}
```

config.ProdConfig.DATABASE

```
ProdConfig.DATABASE = {'engine': 'peewee.SqliteDatabase', 'name': None,
    'pragmas': {'cache_size': -64000, 'foreign_keys': 1,
    'ignore_check_constraints': 0, 'journal_mode': 'wal', 'synchronous': 0}}
```

config.ProdConfig.DEBUG

```
ProdConfig.DEBUG = False
```

config.ProdConfig.DEVELOPMENT

```
ProdConfig.DEVELOPMENT = False
```

config.ProdConfig.FLASK_RESTFUL_PREFIX

```
ProdConfig.FLASK_RESTFUL_PREFIX = '/api'
```

config.ProdConfig.HOME

```
ProdConfig.HOME = '/home/docs'
```

config.ProdConfig.LOGIN_DISABLED

```
ProdConfig.LOGIN_DISABLED = False
```

config.ProdConfig.LOG_DIRECTORY

```
ProdConfig.LOG_DIRECTORY = '/home/docs/checkouts/readthedocs.org/  
user_builds/flask-api/checkouts/latest/log'
```

config.ProdConfig.MAIL_PASSWORD

```
ProdConfig.MAIL_PASSWORD = None
```

config.ProdConfig.MAIL_PORT

```
ProdConfig.MAIL_PORT = None
```

config.ProdConfig.MAIL_SERVER

```
ProdConfig.MAIL_SERVER = None
```

config.ProdConfig.MAIL_USERNAME

```
ProdConfig.MAIL_USERNAME = None
```

config.ProdConfig.MAIL_USE_SSL

```
ProdConfig.MAIL_USE_SSL = False
```

config.ProdConfig.MAIL_USE_TLS

```
ProdConfig.MAIL_USE_TLS = True
```

config.ProdConfig MOCKUP_DIRECTORY

```
ProdConfig.MOCKUP_DIRECTORY = '/home/docs/checkouts/readthedocs.org/  
user_builds/flask-api/checkouts/latest/storage/mockups'
```

config.ProdConfig.RESET_TOKEN_EXPIRES

```
ProdConfig.RESET_TOKEN_EXPIRES = 86400
```

config.ProdConfig.RESTX_ERROR_404_HELP

```
ProdConfig.RESTX_ERROR_404_HELP = False
```

config.ProdConfig.RESTX_MASK_SWAGGER

```
ProdConfig.RESTX_MASK_SWAGGER = False
```

config.ProdConfig.ROOT_DIRECTORY

```
ProdConfig.ROOT_DIRECTORY = '/home/docs/checkouts/readthedocs.org/  
user_builds/flask-api/checkouts/latest'
```

config.ProdConfig.SECRET_KEY

```
ProdConfig.SECRET_KEY = None
```

config.ProdConfig.SECURITY_PASSWORD_HASH

```
ProdConfig.SECURITY_PASSWORD_HASH = 'pbkdf2_sha512'
```

config.ProdConfig.SECURITY_PASSWORD_LENGTH_MIN

```
ProdConfig.SECURITY_PASSWORD_LENGTH_MIN = 8
```

config.ProdConfig.SECURITY_PASSWORD_SALT

```
ProdConfig.SECURITY_PASSWORD_SALT = None
```

config.ProdConfig.SECURITY_TOKEN_AUTHENTICATION_HEADER

```
ProdConfig.SECURITY_TOKEN_AUTHENTICATION_HEADER = 'Authorization'
```

config.ProdConfig.SECURITY_TOKEN_MAX_AGE

```
ProdConfig.SECURITY_TOKEN_MAX_AGE = None
```

config.ProdConfig.SERVER_NAME

```
ProdConfig.SERVER_NAME = None
```

config.ProdConfig.STORAGE_DIRECTORY

```
ProdConfig.STORAGE_DIRECTORY = '/home/docs/checkouts/readthedocs.org/  
user_builds/flask-api/checkouts/latest/storage'
```

config.ProdConfig.SWAGGER_API_URL

```
ProdConfig.SWAGGER_API_URL = 'http://None/static/swagger.yaml'
```

config.ProdConfig.SWAGGER_URL

```
ProdConfig.SWAGGER_URL = '/docs'
```

config.ProdConfig.TESTING

```
ProdConfig.TESTING = False
```

config.ProdConfig.TEST_USER_EMAIL

```
ProdConfig.TEST_USER_EMAIL = None
```

config.ProdConfig.TEST_USER_PASSWORD

```
ProdConfig.TEST_USER_PASSWORD = None
```

config.ProdConfig.accept_content

```
ProdConfig.accept_content = ['json']
```

config.ProdConfig.broker_url

```
ProdConfig.broker_url = 'pyamqp://'
```

config.ProdConfig.enable_utc

```
ProdConfig.enable_utc = True
```


config.ProdConfig.include

```
ProdConfig.include = ['app.celery.tasks']
```

config.ProdConfig.result_backend

```
ProdConfig.result_backend = 'amqp://'
```

config.ProdConfig.result_expires

```
ProdConfig.result_expires = 3600
```

config.ProdConfig.result_extended

```
ProdConfig.result_extended = True
```

config.ProdConfig.result_serializer

```
ProdConfig.result_serializer = 'json'
```

config.ProdConfig.task_always_eager

```
ProdConfig.task_always_eager = False
```

config.ProdConfig.task_default_rate_limit

```
ProdConfig.task_default_rate_limit = 3
```

config.ProdConfig.task_serializer

```
ProdConfig.task_serializer = 'json'
```

config.ProdConfig.task_track_started

```
ProdConfig.task_track_started = True
```

config.ProdConfig.timezone

```
ProdConfig.timezone = 'UTC'
```

config.ProdConfig.worker_log_format

```
ProdConfig.worker_log_format = '%(asctime)s - %(levelname)s -  
%(processName)s - %(message)s'
```

config.ProdConfig.worker_task_log_format

```
ProdConfig.worker_task_log_format = '%(asctime)s - %(levelname)s -  
%(processName)s - %(task_name)s - %(task_id)s - %(message)s'
```

Methods

ProdConfig.__init__()

config.ProdConfig.__init__

```
ProdConfig.__init__()
```

2.4.5 config.TestConfig

class config.TestConfigBases: *config.Config*

Testing configuration options.

Attributes

TestConfig.ALLOWED_CONTENT_TYPES

TestConfig.ALLOWED_MIME_TYPES

TestConfig.DATABASE

TestConfig.DEBUG

TestConfig.DEVELOPMENT

TestConfig.FLASK_RESTFUL_PREFIX

TestConfig.HOME

TestConfig.LOGIN_DISABLED

TestConfig.LOG_DIRECTORY

continues on next page

Table 212 – continued from previous page

<i>TestConfig.MAIL_PASSWORD</i>
<i>TestConfig.MAIL_PORT</i>
<i>TestConfig.MAIL_SERVER</i>
<i>TestConfig.MAIL_USERNAME</i>
<i>TestConfig.MAIL_USE_SSL</i>
<i>TestConfig.MAIL_USE_TLS</i>
<i>TestConfig.MOCKUP_DIRECTORY</i>
<i>TestConfig.RESET_TOKEN_EXPIRES</i>
<i>TestConfig.RESTX_ERROR_404_HELP</i>
<i>TestConfig.RESTX_MASK_SWAGGER</i>
<i>TestConfig.ROOT_DIRECTORY</i>
<i>TestConfig.SECRET_KEY</i>
<i>TestConfig.SECURITY_PASSWORD_HASH</i>
<i>TestConfig.SECURITY_PASSWORD_LENGTH_MIN</i>
<i>TestConfig.SECURITY_PASSWORD_SALT</i>
<i>TestConfig.SECURITY_TOKEN_AUTHENTICATION_HEADER</i>
<i>TestConfig.SECURITY_TOKEN_MAX_AGE</i>
<i>TestConfig.SERVER_NAME</i>
<i>TestConfig.STORAGE_DIRECTORY</i>
<i>TestConfig.SWAGGER_API_URL</i>
<i>TestConfig.SWAGGER_URL</i>
<i>TestConfig.TESTING</i>
<i>TestConfig.TEST_USER_EMAIL</i>
<i>TestConfig.TEST_USER_PASSWORD</i>
<i>TestConfig.accept_content</i>

continues on next page

Table 212 – continued from previous page

<i>TestConfig.broker_url</i>
<i>TestConfig.enable_utc</i>
<i>TestConfig.include</i>
<i>TestConfig.result_backend</i>
<i>TestConfig.result_expires</i>
<i>TestConfig.result_extended</i>
<i>TestConfig.result_serializer</i>
<i>TestConfig.task_always_eager</i>
<i>TestConfig.task_default_rate_limit</i>
<i>TestConfig.task_serializer</i>
<i>TestConfig.task_track_started</i>
<i>TestConfig.timezone</i>
<i>TestConfig.worker_log_format</i>
<i>TestConfig.worker_task_log_format</i>

config.TestConfig.ALLOWED_CONTENT_TYPES

```
TestConfig.ALLOWED_CONTENT_TYPES = {'application/json',  
  'application/octet-stream', 'multipart/form-data'}
```

config.TestConfig.ALLOWED_MIME_TYPES

```
TestConfig.ALLOWED_MIME_TYPES = {'application/pdf',  
  'application/vnd.ms-excel'}
```

config.TestConfig.DATABASE

```
TestConfig.DATABASE = {'engine': 'peewee.SqliteDatabase', 'name':  
  'test.db', 'pragmas': {'cache_size': -64000, 'foreign_keys': 1,  
  'ignore_check_constraints': 0, 'journal_mode': 'wal', 'synchronous': 0}}
```

config.TestConfig.DEBUG

TestConfig.DEBUG = True

config.TestConfig.DEVELOPMENT

TestConfig.DEVELOPMENT = True

config.TestConfig.FLASK_RESTFUL_PREFIX

TestConfig.FLASK_RESTFUL_PREFIX = '/api'

config.TestConfig.HOME

TestConfig.HOME = '/home/docs'

config.TestConfig.LOGIN_DISABLED

TestConfig.LOGIN_DISABLED = False

config.TestConfig.LOG_DIRECTORY

TestConfig.LOG_DIRECTORY = '/home/docs/checkouts/readthedocs.org/
user_builds/flask-api/checkouts/latest/log'

config.TestConfig.MAIL_PASSWORD

TestConfig.MAIL_PASSWORD = None

config.TestConfig.MAIL_PORT

TestConfig.MAIL_PORT = None

config.TestConfig.MAIL_SERVER

TestConfig.MAIL_SERVER = None

config.TestConfig.MAIL_USERNAME

TestConfig.MAIL_USERNAME = None

config.TestConfig.MAIL_USE_SSL

TestConfig.MAIL_USE_SSL = False

config.TestConfig.MAIL_USE_TLS

TestConfig.MAIL_USE_TLS = True

config.TestConfig MOCKUP_DIRECTORY

TestConfig.MOCKUP_DIRECTORY = '/home/docs/checkouts/readthedocs.org/
user_builds/flask-api/checkouts/latest/storage/mockups'

config.TestConfig.RESET_TOKEN_EXPIRES

TestConfig.RESET_TOKEN_EXPIRES = 86400

config.TestConfig.RESTX_ERROR_404_HELP

TestConfig.RESTX_ERROR_404_HELP = False

config.TestConfig.RESTX_MASK_SWAGGER

TestConfig.RESTX_MASK_SWAGGER = False

config.TestConfig.ROOT_DIRECTORY

TestConfig.ROOT_DIRECTORY = '/home/docs/checkouts/readthedocs.org/
user_builds/flask-api/checkouts/latest'

config.TestConfig.SECRET_KEY

TestConfig.SECRET_KEY = None

config.TestConfig.SECURITY_PASSWORD_HASH

TestConfig.SECURITY_PASSWORD_HASH = 'pbkdf2_sha512'

config.TestConfig.SECURITY_PASSWORD_LENGTH_MIN

TestConfig.SECURITY_PASSWORD_LENGTH_MIN = 8

config.TestConfig.SECURITY_PASSWORD_SALT

TestConfig.SECURITY_PASSWORD_SALT = None

config.TestConfig.SECURITY_TOKEN_AUTHENTICATION_HEADER

TestConfig.SECURITY_TOKEN_AUTHENTICATION_HEADER = 'Authorization'

config.TestConfig.SECURITY_TOKEN_MAX_AGE

TestConfig.SECURITY_TOKEN_MAX_AGE = None

config.TestConfig.SERVER_NAME

TestConfig.SERVER_NAME = None

config.TestConfig.STORAGE_DIRECTORY

TestConfig.STORAGE_DIRECTORY = '/home/docs/checkouts/readthedocs.org/
user_builds/flask-api/checkouts/latest/storage/tests'

config.TestConfig.SWAGGER_API_URL

TestConfig.SWAGGER_API_URL = 'http://None/static/swagger.yaml'

config.TestConfig.SWAGGER_URL

TestConfig.SWAGGER_URL = '/docs'

config.TestConfig.TESTING

TestConfig.TESTING = True

config.TestConfig.TEST_USER_EMAIL

TestConfig.TEST_USER_EMAIL = None

config.TestConfig.TEST_USER_PASSWORD

TestConfig.TEST_USER_PASSWORD = None

config.TestConfig.accept_content

TestConfig.accept_content = ['json']

config.TestConfig.broker_url

TestConfig.broker_url = 'pyamqp://'

config.TestConfig.enable_utc

TestConfig.enable_utc = True

config.TestConfig.include

TestConfig.include = ['app.celery.tasks']

config.TestConfig.result_backend

TestConfig.result_backend = 'amqp://'

config.TestConfig.result_expires

TestConfig.result_expires = 3600

config.TestConfig.result_extended

TestConfig.result_extended = True

config.TestConfig.result_serializer

TestConfig.result_serializer = 'json'

config.TestConfig.task_always_eager

TestConfig.task_always_eager = False

config.TestConfig.task_default_rate_limit

```
TestConfig.task_default_rate_limit = 3
```

config.TestConfig.task_serializer

```
TestConfig.task_serializer = 'json'
```

config.TestConfig.task_track_started

```
TestConfig.task_track_started = True
```

config.TestConfig.timezone

```
TestConfig.timezone = 'UTC'
```

config.TestConfig.worker_log_format

```
TestConfig.worker_log_format = '%(asctime)s - %(levelname)s -  
%(processName)s - %(message)s'
```

config.TestConfig.worker_task_log_format

```
TestConfig.worker_task_log_format = '%(asctime)s - %(levelname)s -  
%(processName)s - %(task_name)s - %(task_id)s - %(message)s'
```

Methods

```
TestConfig.__init__()
```

config.TestConfig.__init__

```
TestConfig.__init__()
```

class config.Config

Default configuration options.

```
ALLOWED_CONTENT_TYPES = {'application/json', 'application/octet-stream',  
'multipart/form-data'}
```

```
ALLOWED_MIME_TYPES = {'application/pdf', 'application/vnd.ms-excel'}
```

```
DATABASE = {'engine': 'peewee.SqliteDatabase', 'name': None, 'pragmas':  
{ 'cache_size': -64000, 'foreign_keys': 1, 'ignore_check_constraints': 0,  
'journal_mode': 'wal', 'synchronous': 0 }}
```

```
DEBUG = False
```

```
DEVELOPMENT = False
```

```
FLASK_RESTFUL_PREFIX = '/api'
HOME = '/home/docs'
LOGIN_DISABLED = False
LOG_DIRECTORY =
'/home/docs/checkouts/readthedocs.org/user_builds/flask-api/checkouts/latest/log'
MAIL_PASSWORD = None
MAIL_PORT = None
MAIL_SERVER = None
MAIL_USERNAME = None
MAIL_USE_SSL = False
MAIL_USE_TLS = True
MOCKUP_DIRECTORY = '/home/docs/checkouts/readthedocs.org/user_builds/flask-api/
checkouts/latest/storage/mockups'
RESET_TOKEN_EXPIRES = 86400
RESTX_ERROR_404_HELP = False
RESTX_MASK_SWAGGER = False
ROOT_DIRECTORY =
'/home/docs/checkouts/readthedocs.org/user_builds/flask-api/checkouts/latest'
SECRET_KEY = None
SECURITY_PASSWORD_HASH = 'pbkdf2_sha512'
SECURITY_PASSWORD_LENGTH_MIN = 8
SECURITY_PASSWORD_SALT = None
SECURITY_TOKEN_AUTHENTICATION_HEADER = 'Authorization'
SECURITY_TOKEN_MAX_AGE = None
SERVER_NAME = None
STORAGE_DIRECTORY = '/home/docs/checkouts/readthedocs.org/user_builds/flask-api/
checkouts/latest/storage'
SWAGGER_API_URL = 'http://None/static/swagger.yaml'
SWAGGER_URL = '/docs'
TESTING = False
TEST_USER_EMAIL = None
TEST_USER_PASSWORD = None
accept_content = ['json']
broker_url = 'pyamqp://'
enable_utc = True
include = ['app.celery.tasks']
result_backend = 'amqp://'
```

```

result_expires = 3600
result_extended = True
result_serializer = 'json'
task_always_eager = False
task_default_rate_limit = 3
task_serializer = 'json'
task_track_started = True
timezone = 'UTC'
worker_log_format = '%(asctime)s - %(levelname)s - %(processName)s - %(message)s'
worker_task_log_format = '%(asctime)s - %(levelname)s - %(processName)s -
%(task_name)s - %(task_id)s - %(message)s'

class config.DevConfig
    Development configuration options.

    ALLOWED_CONTENT_TYPES = {'application/json', 'application/octet-stream',
                              'multipart/form-data'}

    ALLOWED_MIME_TYPES = {'application/pdf', 'application/vnd.ms-excel'}

    DATABASE = {'engine': 'peewee.SqliteDatabase', 'name': None, 'pragmas':
                {'cache_size': -64000, 'foreign_keys': 1, 'ignore_check_constraints': 0,
                 'journal_mode': 'wal', 'synchronous': 0}}

    DEBUG = True

    DEVELOPMENT = True

    FLASK_RESTFUL_PREFIX = '/api'

    HOME = '/home/docs'

    LOGIN_DISABLED = False

    LOG_DIRECTORY =
        '/home/docs/checkouts/readthedocs.org/user_builds/flask-api/checkouts/latest/log'

    MAIL_PASSWORD = None

    MAIL_PORT = None

    MAIL_SERVER = None

    MAIL_USERNAME = None

    MAIL_USE_SSL = False

    MAIL_USE_TLS = True

    MOCKUP_DIRECTORY = '/home/docs/checkouts/readthedocs.org/user_builds/flask-api/
checkouts/latest/storage/mockups'

    RESET_TOKEN_EXPIRES = 86400

    RESTX_ERROR_404_HELP = False

    RESTX_MASK_SWAGGER = False

    ROOT_DIRECTORY =
        '/home/docs/checkouts/readthedocs.org/user_builds/flask-api/checkouts/latest'

```

```
SECRET_KEY = None
SECURITY_PASSWORD_HASH = 'pbkdf2_sha512'
SECURITY_PASSWORD_LENGTH_MIN = 8
SECURITY_PASSWORD_SALT = None
SECURITY_TOKEN_AUTHENTICATION_HEADER = 'Authorization'
SECURITY_TOKEN_MAX_AGE = None
SERVER_NAME = None

STORAGE_DIRECTORY = '/home/docs/checkouts/readthedocs.org/user_builds/flask-api/
checkouts/latest/storage'

SWAGGER_API_URL = 'http://None/static/swagger.yaml'
SWAGGER_URL = '/docs'

TESTING = False
TEST_USER_EMAIL = None
TEST_USER_PASSWORD = None
accept_content = ['json']
broker_url = 'pyamqp://'
enable_utc = True
include = ['app.celery.tasks']
result_backend = 'amqp://'
result_expires = 3600
result_extended = True
result_serializer = 'json'
task_always_eager = False
task_default_rate_limit = 3
task_serializer = 'json'
task_track_started = True
timezone = 'UTC'
worker_log_format = '%(asctime)s - %(levelname)s - %(processName)s - %(message)s'
worker_task_log_format = '%(asctime)s - %(levelname)s - %(processName)s -
%(task_name)s - %(task_id)s - %(message)s'

class config.Meta(name: str, bases: tuple, dict: dict)
    Metaclass for updating Config options.

    classmethod _rename_celery_settings(config: type) → None
        Rename old Celery setting names with new ones.
```

References

<https://docs.celeryproject.org/en/latest/userguide/configuration.html#new-lowercase-settings>

mro()

Return a type's method resolution order.

class config.ProdConfig

Production configuration options.

ALLOWED_CONTENT_TYPES = {'application/json', 'application/octet-stream', 'multipart/form-data'}

ALLOWED_MIME_TYPES = {'application/pdf', 'application/vnd.ms-excel'}

DATABASE = {'engine': 'peewee.SqliteDatabase', 'name': None, 'pragmas': {'cache_size': -64000, 'foreign_keys': 1, 'ignore_check_constraints': 0, 'journal_mode': 'wal', 'synchronous': 0}}

DEBUG = False

DEVELOPMENT = False

FLASK_RESTFUL_PREFIX = '/api'

HOME = '/home/docs'

LOGIN_DISABLED = False

LOG_DIRECTORY = '/home/docs/checkouts/readthedocs.org/user_builds/flask-api/checkouts/latest/log'

MAIL_PASSWORD = None

MAIL_PORT = None

MAIL_SERVER = None

MAIL_USERNAME = None

MAIL_USE_SSL = False

MAIL_USE_TLS = True

MOCKUP_DIRECTORY = '/home/docs/checkouts/readthedocs.org/user_builds/flask-api/checkouts/latest/storage/mockups'

RESET_TOKEN_EXPIRES = 86400

RESTX_ERROR_404_HELP = False

RESTX_MASK_SWAGGER = False

ROOT_DIRECTORY = '/home/docs/checkouts/readthedocs.org/user_builds/flask-api/checkouts/latest'

SECRET_KEY = None

SECURITY_PASSWORD_HASH = 'pbkdf2_sha512'

SECURITY_PASSWORD_LENGTH_MIN = 8

SECURITY_PASSWORD_SALT = None

SECURITY_TOKEN_AUTHENTICATION_HEADER = 'Authorization'

SECURITY_TOKEN_MAX_AGE = None

```
SERVER_NAME = None
STORAGE_DIRECTORY = '/home/docs/checkouts/readthedocs.org/user_builds/flask-api/
checkouts/latest/storage'
SWAGGER_API_URL = 'http://None/static/swagger.yaml'
SWAGGER_URL = '/docs'
TESTING = False
TEST_USER_EMAIL = None
TEST_USER_PASSWORD = None
accept_content = ['json']
broker_url = 'pyamqp://'
enable_utc = True
include = ['app.celery.tasks']
result_backend = 'amqp://'
result_expires = 3600
result_extended = True
result_serializer = 'json'
task_always_eager = False
task_default_rate_limit = 3
task_serializer = 'json'
task_track_started = True
timezone = 'UTC'
worker_log_format = '%(asctime)s - %(levelname)s - %(processName)s - %(message)s'
worker_task_log_format = '%(asctime)s - %(levelname)s - %(processName)s -
%(task_name)s - %(task_id)s - %(message)s'
```

```
class config.TestConfig
```

```
    Testing configuration options.
```

```
    ALLOWED_CONTENT_TYPES = {'application/json', 'application/octet-stream',
                             'multipart/form-data'}
```

```
    ALLOWED_MIME_TYPES = {'application/pdf', 'application/vnd.ms-excel'}
```

```
    DATABASE = {'engine': 'peewee.SqliteDatabase', 'name': 'test.db', 'pragmas':
{'cache_size': -64000, 'foreign_keys': 1, 'ignore_check_constraints': 0,
'journal_mode': 'wal', 'synchronous': 0}}
```

```
    DEBUG = True
```

```
    DEVELOPMENT = True
```

```
    FLASK_RESTFUL_PREFIX = '/api'
```

```
    HOME = '/home/docs'
```

```
    LOGIN_DISABLED = False
```

```
LOG_DIRECTORY =
'/home/docs/checkouts/readthedocs.org/user_builds/flask-api/checkouts/latest/log'

MAIL_PASSWORD = None
MAIL_PORT = None
MAIL_SERVER = None
MAIL_USERNAME = None
MAIL_USE_SSL = False
MAIL_USE_TLS = True

MOCKUP_DIRECTORY = '/home/docs/checkouts/readthedocs.org/user_builds/flask-api/
checkouts/latest/storage/mockups'

RESET_TOKEN_EXPIRES = 86400

RESTX_ERROR_404_HELP = False
RESTX_MASK_SWAGGER = False

ROOT_DIRECTORY =
'/home/docs/checkouts/readthedocs.org/user_builds/flask-api/checkouts/latest'

SECRET_KEY = None

SECURITY_PASSWORD_HASH = 'pbkdf2_sha512'
SECURITY_PASSWORD_LENGTH_MIN = 8
SECURITY_PASSWORD_SALT = None
SECURITY_TOKEN_AUTHENTICATION_HEADER = 'Authorization'
SECURITY_TOKEN_MAX_AGE = None

SERVER_NAME = None

STORAGE_DIRECTORY = '/home/docs/checkouts/readthedocs.org/user_builds/flask-api/
checkouts/latest/storage/tests'

SWAGGER_API_URL = 'http://None/static/swagger.yaml'
SWAGGER_URL = '/docs'

TESTING = True

TEST_USER_EMAIL = None
TEST_USER_PASSWORD = None
accept_content = ['json']
broker_url = 'pyamqp://'
enable_utc = True
include = ['app.celery.tasks']
result_backend = 'amqp://'
result_expires = 3600
result_extended = True
result_serializer = 'json'
```

```
task_always_eager = False
task_default_rate_limit = 3
task_serializer = 'json'
task_track_started = True
timezone = 'UTC'
worker_log_format = '%(asctime)s - %(levelname)s - %(processName)s - %(message)s'
worker_task_log_format = '%(asctime)s - %(levelname)s - %(processName)s -
%(task_name)s - %(task_id)s - %(message)s'
```


FLASK COMMAND LINE

Flask command line allow run scripts for managing database, start up task queues, etc.

You don't need to start up the server for running these scripts but you must activate your virtual environment.

CHANGELOG

All notable changes to this project will be documented in this file. See [standard-version](#) for commit guidelines.

4.1 2.0.6 (2021-12-28)

4.2 Bug Fixes

- **pip:** upgrade package versions ([75aff3b](#))

4.3 2.0.5 (2021-12-27)

4.4 Bug Fixes

- **npm:** upgrade package versions ([f62f335](#))

4.4.1 2.0.4 (2021-02-19)

4.5 Bug Fixes

- **users:** add new fs_uniquifier column on users table ([a28be38](#))

4.6 Build System

- **npm:** update node package versions ([6314da3](#))
- **pip:** update flask-security-too version to 4.0.0 ([698acb6](#))
- **pip:** update python package versions ([09755c8](#))

4.6.1 2.0.3 (2021-02-05)

4.7 Bug Fixes

- **celery:** task fields are saving on database when the task is finished ([bc93505](#))

4.8 Code Refactoring

- add new mockups directory in storage directory ([91e642e](#))
- tls and ssl flask mail are getting from environment file ([e548d57](#))

4.8.1 2.0.2 (2021-01-23)

4.9 Bug Fixes

- **celery:** new settings name are not applies on celery 4.4.7 ([f1a0bff](#))
- **middleware:** correct middleware for allowing to let in Swagger ([441b077](#))
- **tasks:** correct task for sending an email with attachments ([43d516c](#))

4.10 Build System

- **pip:** update python dependencies ([8964eed](#))

4.11 Code Refactoring

- **blueprints:** update way to register all blueprints ([b7692b2](#))
- update way to test Celery tasks ([eb3f72d](#))
- **celery:** update way to run Celery ([dee133d](#))
- **factories:** update way to create a model instance from a factory ([ddfcb7d](#))
- **swagger:** update order field format ([ccbfcd4](#))
- **tasks:** exclude internal_filename in Celery tasks ([53dcf6d](#))

4.11.1 2.0.1 (2021-01-05)

4.12 Bug Fixes

- **tasks:** correct output data on serializers and swagger ([d5c050a](#))

4.13 Build System

- **npm:** update npm dependencies for fixing a vulnerability found ([15ee5de](#))
- **pip:** update Python packages ([11676b6](#))

4.14 Code Refactoring

- **blueprints:** import dynamic way ([30ee0a0](#))
- **blueprints:** remove logging module from all blueprints ([a51709e](#))
- **db:** models available could be import from init module ([2361e0e](#))
- **db:** move user_roles model to its own module ([650eb39](#))
- **exceptions:** add error handler to marshmallow validation error class ([10bb664](#))
- **exceptions:** correct error handler to marshmallow validation error class ([ba30805](#))
- **managers:** add new logic for managing database queries through database models ([758e077](#))
- **serializers:** correct user email validation ([2d733cf](#))
- **serializers:** move app.marshmallow_schema.py to serializers package ([47f75e1](#))
- **serializers:** upgrade validation fields ([24ac198](#))
- **services:** add new logic for managing business logic to auth ([3c969b5](#))
- **services:** add new logic for managing business logic to documents ([1ce7b96](#))
- **services:** add new logic for managing business logic to roles ([7ff4c1](#))
- **services:** add new logic for managing business logic to tasks ([079741d](#))
- **services:** add new logic for managing business logic to users ([572f37f](#))
- **swagger:** move app.utils.swagger_models to app.swagger package ([ef4dd87](#))
- **swagger:** update document swagger models ([7015c61](#))
- **swagger:** update role swagger models ([d618021](#))
- **swagger:** update user swagger models ([766e697](#))
- **tasks:** add suffix to task names ([998a56d](#))
- **utils:** create new modules to constants and request query operators ([cbf12c9](#))

4.14.1 2.0.0 (2020-10-27)

4.15 BREAKING CHANGES

- **order** field in search requests is a list of dicts.

4.16 Features

- **factories:** add prevent code for checking if a given model is registered as factory ([1605a01](#))
- **shell:** import Factory class to Flask interactive shell ([8cf9c8e](#))

4.17 Code Refactoring

- replace cerberus to flask-mashmallow validation ([7552d5c](#))
- **celery:** replace old Celery setting names with new ones ([15e0c03](#))
- **celery:** update way to set FLASK_CONFIG value on Flask command ([a865548](#))

4.18 Build System

- add .versionrc that shows build/perf/refactor/revert ([0017d66](#))
- add sphinx-click configuration to Sphinx and create new file for showing Click documentation ([bb41b7b](#))
- add sphinx-click for showing Click documentation in Sphinx ([c0a8f55](#))
- **pip:** remove cerberus package ([4a4fe72](#))
- **pip:** split python packages in two requirements local and production ([f39a2a5](#))

4.18.1 1.4.1 (2020-10-07)

4.19 Bug Fixes

- **celery:** correct problem when start Celery ([52ad2fb](#)), closes #3

4.19.1 1.4.0 (2020-10-04)

4.20 Features

- **celery:** add task for exporting several files ([ca8355f](#))
- **documentation:** add sphinx integration ([8c313fd](#))

4.20.1 1.3.0 (2020-09-20)

4.21 Features

- **swagger:** add Swagger full integration ([1eaf8d8](#))

4.21.1 1.2.0 (2020-09-18)

4.22 BREAKING CHANGES

- install/update Node.js and Python libraries

4.23 build

- update Node.js and Python packages ([b7416cc](#))

4.23.1 1.1.0 (2020-05-31)

4.24 Features

- **security:** add role-based authorization ([345b57e](#))
- add advanced search in documents, roles and users ([8fce3e3](#))
- add marshmallow package integration ([a8b647e](#))
- add Swagger integration ([dc6ace4](#))

4.25 Refactor

- replace HTTP exceptions to Werkzeug HTTP Exceptions ([31e5606](#))
- move Word and Excel celery tasks to them own modules ([00e42e5](#))

4.26 Docs

- docs: add installation project guide ([b915d31](#))

4.26.1 1.0.0 (2020-05-17)

4.27 BREAKING CHANGES

- **pip:** update Python dependencies

4.28 Features

- **celery:** add basic installation ([147dd2c](#))
- **db:** add peewee migrations ([231696c](#))
- **documents:** add document logic ([1eb7ec1](#))
- **emails:** add send emails after creation an user ([7c2cfe0](#))
- **log:** add support for logrotate ([09925e1](#))
- **users:** add created_by column in user model ([8a3d013](#))
- **users:** add Excel and PDF users export to background processes ([781e091](#))
- **users:** add recovery password feature ([e1e916e](#))

4.29 Build

- **pip:** update requirements.txt ([6193153](#))

4.29.1 0.8.0 (2020-04-29)

4.30 Features

- **roles:** add role logic ([d7a0535](#))
- **security:** add jwt authentication ([fb51089](#))
- **users:** add role model integration to user model ([69bc124](#))
- **users:** add user get endpoint ([018b965](#))

4.30.1 0.7.0 (2020-04-23)

4.31 Features

- **doc:** add standard-version NodeJS package ([c1b2cb3](#))

4.31.1 0.6.1 (2020-04-23)

4.32 BREAKING CHANGES

- update python dependencies

4.33 Features

- **db:** added script for creating database tables ([c14b566](#))
- **logging:** added logging configuration ([297b9c3](#))
- **seeders:** added user seeder ([e78b4c4](#))
- **tests:** add tests and code coverage ([17317b7](#))
- **validation-requests:** add validation requests with cerberus ([a5beed6](#))

4.34 Bug Fixes

- **commitizen:** fixed problem with the process of commitizen tags ([1d3677d](#))
- **docs_to_pdf:** fixed problem about convert a docx file to a pdf file with uWSGI ([aabbcc2d](#))
- **peewee:** fixed problem about a connection already opened error ([6279470](#))
- **peewee:** problem with database connection already opened ([e6c07c9](#))
- **users:** update user endpoint cannot update data ([9dfc4cc](#))
- request search fields in search users, export PDF and export Excel endpoints ([2ae7ab7](#))

4.35 Build

- update requirements.txt ([f783e78](#))
- update requirements.txt ([b6378ba](#))

NOTE

If you find any bugs, odd behavior, or have an idea for a new feature please don't hesitate to on GitHub.

INDICES AND TABLES

- `genindex`
- `modindex`
- `search`

PYTHON MODULE INDEX

a

- app, 8
- app.blueprints, 8
- app.blueprints.auth, 9
- app.blueprints.base, 23
- app.blueprints.documents, 29
- app.blueprints.roles, 42
- app.blueprints.tasks, 55
- app.blueprints.users, 61
- app.celery, 84
- app.celery.excel, 85
- app.celery.excel.tasks, 85
- app.celery.tasks, 86
- app.celery.tests, 86
- app.celery.tests.tasks, 86
- app.celery.word, 87
- app.celery.word.tasks, 87
- app.exceptions, 139
- app.extensions, 140
- app.managers, 141
- app.managers.base, 141
- app.managers.document, 143
- app.managers.role, 145
- app.managers.user, 146
- app.middleware, 149
- app.models, 150
- app.models.base, 151
- app.models.document, 160
- app.models.role, 171
- app.models.user, 182
- app.models.user_roles, 200
- app.serializers, 200
- app.serializers.auth, 201
- app.serializers.core, 225
- app.serializers.document, 254
- app.serializers.role, 279
- app.serializers.user, 297
- app.services, 326
- app.services.auth, 327
- app.services.base, 328
- app.services.document, 330
- app.services.role, 331

- app.services.task, 333
- app.services.user, 335
- app.swagger, 336
- app.swagger.auth, 337
- app.swagger.core, 337
- app.swagger.document, 337
- app.swagger.role, 337
- app.swagger.user, 337
- app.utils, 337
- app.utils.constants, 337
- app.utils.decorators, 338
- app.utils.file_storage, 338
- app.utils.libreoffice, 340
- app.utils.request_query_operator, 341

c

- config, 399

d

- database, 348
- database.factories, 348
- database.migrations, 352
- database.migrations.aaa_add_genre_column_on_user_table, 353
- database.migrations.aab_add_created_by_column_on_user_table, 354
- database.migrations.aac_create_documents_table, 355
- database.migrations.aad_create_user_roles_table, 356
- database.migrations.aaf_remove_role_slug_column, 359
- database.migrations.aag_add_fs_uniquifier_column_on_users, 362
- database.seeds, 373
- database.seeds.document_seeder, 373
- database.seeds.role_seeder, 374
- database.seeds.user_seeder, 375

t

- tests, 377
- tests.blueprints, 379

tests.blueprints.test_auth, 379
tests.blueprints.test_base, 380
tests.blueprints.test_documents, 381
tests.blueprints.test_roles, 383
tests.blueprints.test_tasks, 384
tests.blueprints.test_users, 385
tests.celery, 387
tests.celery.test_celery, 387
tests.celery.test_excel, 388
tests.celery.test_tasks, 388
tests.celery.test_word, 389
tests.conftest, 390
tests.custom_flask_client, 391
tests.test_config, 397
tests.test_db, 397
tests.test_mail, 397
tests.test_middleware, 398

INDEX

Symbols

Symbols

<code>__CHECK_ATTRIBUTE</code> (<i>app.serializers.core.TimestampField attribute</i>), 240	<code>__init__</code> () (<i>app.blueprints.documents.SearchDocumentResource method</i>), 39
<code>__CHECK_ATTRIBUTE</code> (<i>app.serializers.role.RoleName attribute</i>), 288	<code>__init__</code> () (<i>app.blueprints.roles.NewRoleResource method</i>), 44
<code>__CHECK_ATTRIBUTE</code> (<i>app.serializers.user.VerifyRoleId attribute</i>), 324	<code>__init__</code> () (<i>app.blueprints.roles.RoleBaseResource method</i>), 47
<code>__OldRole</code> (<i>class in database.migrations.aaf_remove_role_slug_column</i>), 360	<code>__init__</code> () (<i>app.blueprints.roles.RoleResource method</i>), 49
<code>__OldUser</code> (<i>class in database.migrations.aad_create_user_roles_table</i>), 356	<code>__init__</code> () (<i>app.blueprints.roles.RolesSearchResource method</i>), 52
<code>__SearchOrderSerializer</code> (<i>class in app.serializers.core</i>), 242	<code>__init__</code> () (<i>app.blueprints.tasks.TaskResource method</i>), 57
<code>__SearchOrderSerializer.Meta</code> (<i>class in app.serializers.core</i>), 242	<code>__init__</code> () (<i>app.blueprints.tasks.TaskStatusResource method</i>), 59
<code>__SearchValueSerializer</code> (<i>class in app.serializers.core</i>), 248	<code>__init__</code> () (<i>app.blueprints.users.ExportUsersExcelAndWordResource method</i>), 63
<code>__SearchValueSerializer.Meta</code> (<i>class in app.serializers.core</i>), 248	<code>__init__</code> () (<i>app.blueprints.users.ExportUsersExcelResource method</i>), 66
<code>__check_exists_factory</code> () (<i>database.factories.Factory method</i>), 352	<code>__init__</code> () (<i>app.blueprints.users.ExportUsersWordResource method</i>), 69
<code>__current_module</code> (<i>database.factories.Factory attribute</i>), 352	<code>__init__</code> () (<i>app.blueprints.users.NewUserResource method</i>), 71
<code>__init__</code> () (<i>app.blueprints.auth.AuthBaseResource method</i>), 11	<code>__init__</code> () (<i>app.blueprints.users.UserBaseResource method</i>), 74
<code>__init__</code> () (<i>app.blueprints.auth.AuthUserLoginResource method</i>), 13	<code>__init__</code> () (<i>app.blueprints.users.UserResource method</i>), 76
<code>__init__</code> () (<i>app.blueprints.auth.AuthUserLogoutResource method</i>), 15	<code>__init__</code> () (<i>app.blueprints.users.UsersSearchResource method</i>), 79
<code>__init__</code> () (<i>app.blueprints.auth.RequestResetPasswordResource method</i>), 17	<code>__init__</code> () (<i>app.celery.ContextTask method</i>), 94
<code>__init__</code> () (<i>app.blueprints.auth.ResetPasswordResource method</i>), 19	<code>__init__</code> () (<i>app.celery.MyCelery method</i>), 111
<code>__init__</code> () (<i>app.blueprints.base.BaseResource method</i>), 25	<code>__init__</code> () (<i>app.managers.base.BaseManager method</i>), 142
<code>__init__</code> () (<i>app.blueprints.base.WelcomeResource method</i>), 27	<code>__init__</code> () (<i>app.managers.document.DocumentManager method</i>), 144
<code>__init__</code> () (<i>app.blueprints.documents.DocumentBaseResource method</i>), 31	<code>__init__</code> () (<i>app.managers.role.RoleManager method</i>), 145
<code>__init__</code> () (<i>app.blueprints.documents.DocumentResource method</i>), 33	<code>__init__</code> () (<i>app.managers.user.UserManager method</i>), 147
<code>__init__</code> () (<i>app.blueprints.documents.NewDocumentResource method</i>), 35	<code>__init__</code> () (<i>app.middleware.Middleware method</i>), 149
	<code>__init__</code> () (<i>app.models.base.Base method</i>), 153
	<code>__init__</code> () (<i>app.models.document.Document method</i>), 155

[__init__\(\)](#) ([app.models.role.Role](#) method), 175
[__init__\(\)](#) ([app.models.user.User](#) method), 188
[__init__\(\)](#) ([app.serializers.auth.AuthUserConfirmResetPasswordSerializer](#) method), 203
[__init__\(\)](#) ([app.serializers.auth.AuthUserLoginSerializer](#) method), 209
[__init__\(\)](#) ([app.serializers.core.SearchSerializer](#) method), 227
[__init__\(\)](#) ([app.serializers.core.TimestampField](#) method), 233
[__init__\(\)](#) ([app.serializers.document.DocumentAttachmentSerializer](#) method), 257
[__init__\(\)](#) ([app.serializers.document.DocumentSerializer](#) method), 262
[__init__\(\)](#) ([app.serializers.role.RoleName](#) method), 281
[__init__\(\)](#) ([app.serializers.role.RoleSerializer](#) method), 284
[__init__\(\)](#) ([app.serializers.user.UserExportWordSerializer](#) method), 299
[__init__\(\)](#) ([app.serializers.user.UserSerializer](#) method), 305
[__init__\(\)](#) ([app.serializers.user.VerifyRoleId](#) method), 310
[__init__\(\)](#) ([app.services.auth.AuthService](#) method), 327
[__init__\(\)](#) ([app.services.base.BaseService](#) method), 329
[__init__\(\)](#) ([app.services.document.DocumentService](#) method), 330
[__init__\(\)](#) ([app.services.role.RoleService](#) method), 332
[__init__\(\)](#) ([app.services.task.TaskService](#) method), 334
[__init__\(\)](#) ([app.services.user.UserService](#) method), 335
[__init__\(\)](#) ([app.utils.file_storage.FileStorage](#) method), 339
[__init__\(\)](#) ([app.utils.request_query_operator.Helper](#) method), 342
[__init__\(\)](#) ([app.utils.request_query_operator.RequestQueryOperator](#) method), 343
[__init__\(\)](#) ([config.Config](#) method), 407
[__init__\(\)](#) ([config.DevConfig](#) method), 414
[__init__\(\)](#) ([config.Meta](#) method), 414
[__init__\(\)](#) ([config.ProdConfig](#) method), 422
[__init__\(\)](#) ([config.TestConfig](#) method), 429
[__init__\(\)](#) ([database.factories.Factory](#) method), 350
[__init__\(\)](#) ([database.migrations.Migration](#) method), 366
[__init__\(\)](#) ([database.migrations.aaa_add_genre_column](#) method), 353
[__init__\(\)](#) ([database.migrations.aab_add_created_by_column](#) method), 354
[__init__\(\)](#) ([database.migrations.aac_create_documents_table](#) method), 355
[__init__\(\)](#) ([database.migrations.aad_create_user_roles_table](#) method), 356
[__init__\(\)](#) ([database.migrations.aaf_remove_role_slug_column](#) method), 359
[__init__\(\)](#) ([database.migrations.aag_add_fs_uniquifier_column_on_user_roles_table](#) method), 362
[__init__\(\)](#) ([database.seeds.document_seeder.DocumentSeeder](#) method), 374
[__init__\(\)](#) ([database.seeds.role_seeder.RoleSeeder](#) method), 375
[__init__\(\)](#) ([database.seeds.user_seeder.UserSeeder](#) method), 376
[__init__\(\)](#) ([tests.custom_flask_client.CustomFlaskClient](#) method), 392
[__models__](#) ([database.factories.Factory](#) attribute), 352
[__user__](#) ([app.serializers.auth.AuthUserLoginSerializer](#) attribute), 220
[_acquire_connection\(\)](#) ([app.celery.MyCelery](#) method), 130
[_add_excel_autofilter\(\)](#) ([app.celery.excel.tasks](#) module), 85
[_add_foreign_key_constraint_users_table\(\)](#) ([database.migrations.aad_create_user_roles_table.CreateUserRoleTable](#) static method), 356
[_add_periodic_task\(\)](#) ([app.celery.MyCelery](#) method), 130
[_add_table_column_names\(\)](#) ([app.celery.word.tasks](#) module), 87
[_add_table_user_data\(\)](#) ([app.celery.word.tasks](#) module), 87
[_add_unique_constraint_roles_table\(\)](#) ([database.migrations.aaf_remove_role_slug_column.RemoveRoleSlugColumn](#) static method), 359
[_adjust_each_column_width\(\)](#) ([app.celery.excel.tasks](#) module), 85
[_after_fork\(\)](#) ([app.celery.MyCelery](#) method), 130
[_after_fork_registered](#) ([app.celery.MyCelery](#) attribute), 130
[_app](#) ([app.celery.ContextTask](#) attribute), 120
[_autodiscover_tasks\(\)](#) ([app.celery.MyCelery](#) method), 130
[_autodiscover_tasks_from_fixups\(\)](#) ([app.celery.MyCelery](#) method), 130
[_autodiscover_tasks_from_names\(\)](#) ([app.celery.MyCelery](#) method), 130
[_backend](#) ([app.celery.ContextTask](#) attribute), 120
[_bind_field\(\)](#) ([app.serializers.auth.AuthUserConfirmResetPasswordSerializer](#) method), 214
[_bind_field\(\)](#) ([app.serializers.auth.AuthUserLoginSerializer](#) method), 220
[_bind_or_create\(\)](#) ([app.celery.ContextTask](#) attribute), 120

`method`), 235
`_bind_field()` (`app.serializers.core._SearchOrderSerializer` `method`), 243
`_bind_field()` (`app.serializers.core._SearchValueSerializer` `method`), 249
`_bind_field()` (`app.serializers.document.DocumentAttachmentSerializer` `method`), 267
`_bind_field()` (`app.serializers.document.DocumentSerializer` `method`), 272
`_bind_field()` (`app.serializers.role.RoleSerializer` `method`), 291
`_bind_field()` (`app.serializers.user.UserExportWordSerializer` `method`), 312
`_bind_field()` (`app.serializers.user.UserSerializer` `method`), 318
`_bind_to_schema()` (`app.serializers.core.TimestampField` `method`), 240
`_bind_to_schema()` (`app.serializers.role.RoleName` `method`), 288
`_bind_to_schema()` (`app.serializers.user.VerifyRoleId` `method`), 324
`_call_and_store()` (`app.serializers.auth.AuthUserConfirmResetPasswordSerializer` `static method`), 214
`_call_and_store()` (`app.serializers.auth.AuthUserLoginSerializer` `static method`), 220
`_call_and_store()` (`app.serializers.core.SearchSerializer` `static method`), 235
`_call_and_store()` (`app.serializers.core._SearchOrderSerializer` `static method`), 243
`_call_and_store()` (`app.serializers.core._SearchValueSerializer` `static method`), 249
`_call_and_store()` (`app.serializers.document.DocumentAttachmentSerializer` `static method`), 267
`_call_and_store()` (`app.serializers.document.DocumentSerializer` `static method`), 272
`_call_and_store()` (`app.serializers.role.RoleSerializer` `static method`), 291
`_call_and_store()` (`app.serializers.user.UserExportWordSerializer` `static method`), 313
`_call_and_store()` (`app.serializers.user.UserSerializer` `static method`), 318
`_canvas` (`app.celery.MyCelery` `attribute`), 130
`_coerce` (`app.models.base.Base` `attribute`), 158
`_coerce` (`app.models.document.Document` `attribute`), 169
`_coerce` (`app.models.role.Role` `attribute`), 180
`_coerce` (`app.models.user.User` `attribute`), 196
`_coerce` (`database.migrations.Migration` `attribute`), 371
`_coerce` (`database.migrations.aad_create_user_roles_table._OldUser` `attribute`), 357
`_coerce` (`database.migrations.aaf_remove_role_slug_column._OldRole` `attribute`), 360
`_conf` (`app.celery.MyCelery` `attribute`), 130
`_connection()` (`app.celery.MyCelery` `method`), 130
`_create_admin_role()` (`database.seeds.role_seeder.RoleSeeder` `static method`), 375
`_create_admin_user()` (`database.seeds.user_seeder.UserSeeder` `static method`), 376
`_create_task_record()` (`in module tests.blueprints.test_tasks`), 384
`_create_team_leader()` (`database.seeds.role_seeder.RoleSeeder` `static method`), 375
`_create_worker_role()` (`database.seeds.role_seeder.RoleSeeder` `static method`), 375
`_creation_index` (`app.serializers.core.TimestampField` `attribute`), 240
`_creation_index` (`app.serializers.role.RoleName` `attribute`), 288
`_creation_index` (`app.serializers.user.VerifyRoleId` `attribute`), 324
`_declared_fields` (`app.serializers.auth.AuthUserConfirmResetPasswordSerializer` `attribute`), 214
`_declared_fields` (`app.serializers.auth.AuthUserLoginSerializer` `attribute`), 220
`_declared_fields` (`app.serializers.core.SearchSerializer` `attribute`), 235
`_declared_fields` (`app.serializers.core._SearchOrderSerializer` `attribute`), 244
`_declared_fields` (`app.serializers.core._SearchValueSerializer` `attribute`), 249
`_declared_fields` (`app.serializers.document.DocumentAttachmentSerializer` `attribute`), 268
`_declared_fields` (`app.serializers.document.DocumentSerializer` `attribute`), 273
`_declared_fields` (`app.serializers.role.RoleSerializer` `attribute`), 291
`_declared_fields` (`app.serializers.user.UserExportWordSerializer` `attribute`), 313
`_declared_fields` (`app.serializers.user.UserSerializer` `attribute`), 318
`_default_error_messages` (`app.serializers.auth.AuthUserConfirmResetPasswordSerializer` `attribute`), 215
`_default_error_messages` (`app.serializers.auth.AuthUserLoginSerializer` `attribute`), 221
`_default_error_messages` (`app.serializers.core.SearchSerializer` `attribute`), 236
`_default_error_messages` (`app.serializers.core._SearchOrderSerializer` `attribute`), 244
`_default_error_messages` (`app.serializers.core._SearchValueSerializer` `attribute`), 249

attribute), 250
 _default_error_messages
 (app.serializers.document.DocumentAttachmentSerializer attribute), 268
 _default_error_messages
 (app.serializers.document.DocumentSerializer attribute), 275
 _default_error_messages
 (app.serializers.role.RoleSerializer attribute), 292
 _default_error_messages
 (app.serializers.user.UserExportWordSerializer attribute), 313
 _default_error_messages
 (app.serializers.user.UserSerializer attribute), 320
 _default_request (app.celery.ContextTask attribute), 120
 _deserialize() (app.serializers.auth.AuthUserConfirmResetPasswordSerializer method), 215
 _deserialize() (app.serializers.auth.AuthUserLoginSerializer method), 221
 _deserialize() (app.serializers.core.SearchSerializer method), 236
 _deserialize() (app.serializers.core.TimestampField method), 240
 _deserialize() (app.serializers.core._SearchOrderSerializer method), 244
 _deserialize() (app.serializers.core._SearchValueSerializer method), 250
 _deserialize() (app.serializers.document.DocumentAttachmentSerializer method), 268
 _deserialize() (app.serializers.document.DocumentSerializer method), 275
 _deserialize() (app.serializers.role.RoleName method), 288
 _deserialize() (app.serializers.role.RoleSerializer method), 292
 _deserialize() (app.serializers.user.UserExportWordSerializer method), 313
 _deserialize() (app.serializers.user.UserSerializer method), 320
 _deserialize() (app.serializers.user.VerifyRoleId method), 324
 _do_load() (app.serializers.auth.AuthUserConfirmResetPasswordSerializer method), 215
 _do_load() (app.serializers.auth.AuthUserLoginSerializer method), 221
 _do_load() (app.serializers.core.SearchSerializer method), 236
 _do_load() (app.serializers.core._SearchOrderSerializer method), 244
 _do_load() (app.serializers.core._SearchValueSerializer method), 250
 _do_load() (app.serializers.document.DocumentAttachmentSerializer method), 268
 _do_load() (app.serializers.document.DocumentSerializer method), 275
 _do_load() (app.serializers.role.RoleSerializer method), 293
 _do_load() (app.serializers.user.UserExportWordSerializer method), 313
 _do_load() (app.serializers.user.UserSerializer method), 320
 _drop_foreign_key_constraint_users_table()
 (database.migrations.aad_create_user_roles_table.CreateUserRoleTable static method), 356
 _drop_unique_constraint_roles_table()
 (database.migrations.aaf_remove_role_slug_column.RemoveRoleSlugColumn static method), 359
 _ensure_after_fork() (app.celery.MyCelery method), 130
 _extract_options(app.celery.ContextTask attribute), 121
 _exists_column() (database.migrations.aaa_add_genre_column_on_user_genre static method), 353
 _exists_column() (database.migrations.aab_add_created_by_column_on_user static method), 354
 _exists_column() (database.migrations.aaf_remove_role_slug_column.RemoveRoleSlugColumn static method), 359
 _exists_column() (database.migrations.aag_add_fs_uniquifier_column_on_file static method), 363
 _exists_table() (database.migrations.aac_create_documents_table.CreateDocumentsTable static method), 355
 _exists_table() (database.migrations.aad_create_user_roles_table.CreateUserRoleTable static method), 356
 _finalize_pending_conf() (app.celery.MyCelery method), 130
 _fixups (app.celery.MyCelery attribute), 130
 _get_app() (app.celery.ContextTask class method), 121
 _get_backend() (app.celery.MyCelery method), 130
 _get_default_loader() (app.celery.MyCelery method), 131
 _get_excel_column_names() (in module app.celery.excel.tasks), 85
 _get_excel_user_data() (in module app.celery.excel.tasks), 85
 _get_exec_options() (app.celery.ContextTask method), 121
 _get_executors() (app.celery.ContextTask method), 121
 _get_user_data() (in module app.celery.excel.tasks), 85
 _get_user_data() (in module app.celery.word.tasks), 87
 _handle_validation_error_exception() (in module app.exceptions), 140
 _has_processors() (app.serializers.auth.AuthUserConfirmResetPasswordSerializer method), 215
 _has_processors() (app.serializers.auth.AuthUserLoginSerializer

method), 221
 _has_processors() (app.serializers.core.SearchSerializer_init_fields() method), 237
 _has_processors() (app.serializers.core._SearchOrderSerializer method), 245
 _has_processors() (app.serializers.core._SearchValueSerializer method), 251
 _has_processors() (app.serializers.document.DocumentAttachmentSerializer method), 268
 _has_processors() (app.serializers.document.DocumentSerializer method), 275
 _has_processors() (app.serializers.role.RoleSerializer method), 293
 _has_processors() (app.serializers.user.UserExportWordSerializer method), 314
 _has_processors() (app.serializers.user.UserSerializer method), 320
 _hooks (app.serializers.auth.AuthUserConfirmResetPasswordSerializer attribute), 215
 _hooks (app.serializers.auth.AuthUserLoginSerializer attribute), 222
 _hooks (app.serializers.core.SearchSerializer attribute), 237
 _hooks (app.serializers.core._SearchOrderSerializer attribute), 245
 _hooks (app.serializers.core._SearchValueSerializer attribute), 251
 _hooks (app.serializers.document.DocumentAttachmentSerializer attribute), 268
 _hooks (app.serializers.document.DocumentSerializer attribute), 275
 _hooks (app.serializers.role.RoleSerializer attribute), 293
 _hooks (app.serializers.user.UserExportWordSerializer attribute), 314
 _hooks (app.serializers.user.UserSerializer attribute), 320
 _init_fields() (app.serializers.auth.AuthUserConfirmResetPasswordSerializer method), 216
 _init_fields() (app.serializers.auth.AuthUserLoginSerializer method), 222
 _init_fields() (app.serializers.core.SearchSerializer method), 237
 _init_fields() (app.serializers.core._SearchOrderSerializer method), 245
 _init_fields() (app.serializers.core._SearchValueSerializer method), 251
 _init_fields() (app.serializers.document.DocumentAttachmentSerializer method), 268
 _init_fields() (app.serializers.document.DocumentSerializer method), 275
 _init_fields() (app.serializers.role.RoleSerializer method), 293
 _init_fields() (app.serializers.user.UserExportWordSerializer method), 314
 _init_fields() (app.serializers.user.UserSerializer method), 320
 _invoke_dump_processors() (app.serializers.auth.AuthUserConfirmResetPasswordSerializer method), 216
 _invoke_dump_processors() (app.serializers.auth.AuthUserLoginSerializer method), 222
 _invoke_dump_processors() (app.serializers.core.SearchSerializer method), 237
 _invoke_dump_processors() (app.serializers.core._SearchOrderSerializer method), 245
 _invoke_dump_processors() (app.serializers.core._SearchValueSerializer method), 251
 _invoke_dump_processors() (app.serializers.document.DocumentAttachmentSerializer method), 269
 _invoke_dump_processors() (app.serializers.document.DocumentSerializer method), 275
 _invoke_dump_processors() (app.serializers.role.RoleSerializer method), 293
 _invoke_dump_processors() (app.serializers.user.UserExportWordSerializer method), 314
 _invoke_dump_processors() (app.serializers.user.UserSerializer method), 320
 _invoke_field_validators() (app.serializers.auth.AuthUserConfirmResetPasswordSerializer method), 216
 _invoke_field_validators() (app.serializers.auth.AuthUserLoginSerializer method), 222
 _invoke_field_validators() (app.serializers.core.SearchSerializer method), 237
 _invoke_field_validators() (app.serializers.core._SearchOrderSerializer method), 245
 _invoke_field_validators() (app.serializers.core._SearchValueSerializer method), 251
 _invoke_field_validators() (app.serializers.document.DocumentAttachmentSerializer method), 269
 _invoke_field_validators() (app.serializers.document.DocumentSerializer method), 275
 _invoke_field_validators() (app.serializers.role.RoleSerializer method), 293
 _invoke_field_validators() (app.serializers.user.UserExportWordSerializer method), 314
 _invoke_field_validators() (app.serializers.user.UserSerializer method), 320

`method)`, 275
`_invoke_field_validators()`
 (`app.serializers.role.RoleSerializer` `method`), 293
`_invoke_field_validators()`
 (`app.serializers.user.UserExportWordSerializer` `method`), 314
`_invoke_field_validators()`
 (`app.serializers.user.UserSerializer` `method`), 320
`_invoke_load_processors()`
 (`app.serializers.auth.AuthUserConfirmResetPasswordSerializer` `method`), 216
`_invoke_load_processors()`
 (`app.serializers.auth.AuthUserLoginSerializer` `method`), 222
`_invoke_load_processors()`
 (`app.serializers.core.SearchSerializer` `method`), 237
`_invoke_load_processors()`
 (`app.serializers.core._SearchOrderSerializer` `method`), 245
`_invoke_load_processors()`
 (`app.serializers.core._SearchValueSerializer` `method`), 251
`_invoke_load_processors()`
 (`app.serializers.document.DocumentAttachmentSerializer` `method`), 269
`_invoke_load_processors()`
 (`app.serializers.document.DocumentSerializer` `method`), 275
`_invoke_load_processors()`
 (`app.serializers.role.RoleSerializer` `method`), 293
`_invoke_load_processors()`
 (`app.serializers.user.UserExportWordSerializer` `method`), 314
`_invoke_load_processors()`
 (`app.serializers.user.UserSerializer` `method`), 320
`_invoke_processors()`
 (`app.serializers.auth.AuthUserConfirmResetPasswordSerializer` `method`), 216
`_invoke_processors()`
 (`app.serializers.auth.AuthUserLoginSerializer` `method`), 222
`_invoke_processors()`
 (`app.serializers.core.SearchSerializer` `method`), 237
`_invoke_processors()`
 (`app.serializers.core._SearchOrderSerializer` `method`), 245
`_invoke_processors()`
 (`app.serializers.core._SearchValueSerializer` `method`), 251
`_invoke_processors()`
 (`app.serializers.document.DocumentAttachmentSerializer` `method`), 269
`_invoke_processors()`
 (`app.serializers.document.DocumentSerializer` `method`), 276
`_invoke_processors()`
 (`app.serializers.role.RoleSerializer` `method`), 293
`_invoke_processors()`
 (`app.serializers.user.UserExportWordSerializer` `method`), 314
`_invoke_processors()`
 (`app.serializers.user.UserSerializer` `method`), 320
`_load_config()` (`app.celery.MyCelery` `method`), 131
`_local` (`app.celery.MyCelery` `attribute`), 131
`_meta` (`app.models.base.Base` `attribute`), 158
`_meta` (`app.models.document.Document` `attribute`), 169
`_meta` (`app.models.role.Role` `attribute`), 180
`_meta` (`app.models.user.User` `attribute`), 196
`_meta` (`database.migrations.Migration` `attribute`), 371
`_meta` (`database.migrations.aad_create_user_roles_table._OldUser`

attribute), 357
 _meta (database.migrations.aaf_remove_role_slug_column._OldRole attribute), 360
 _normalize_data() (app.models.base.Base class method), 158
 _normalize_data() (app.models.document.Document class method), 169
 _normalize_data() (app.models.role.Role class method), 180
 _normalize_data() (app.models.user.User class method), 196
 _normalize_data() (database.migrations.Migration class method), 371
 _normalize_data() (database.migrations.aad_create_user_roles_table._OldUser class method), 357
 _normalize_data() (database.migrations.aaf_remove_role_slug_column._OldRole class method), 360
 _normalize_nested_options() (app.serializers.auth.AuthUserConfirmResetPasswordSerializer method), 216
 _normalize_nested_options() (app.serializers.auth.AuthUserLoginSerializer method), 222
 _normalize_nested_options() (app.serializers.core.SearchSerializer method), 237
 _normalize_nested_options() (app.serializers.core._SearchOrderSerializer method), 245
 _normalize_nested_options() (app.serializers.core._SearchValueSerializer method), 251
 _normalize_nested_options() (app.serializers.document.DocumentAttachmentSerializer method), 269
 _normalize_nested_options() (app.serializers.document.DocumentSerializer method), 276
 _normalize_nested_options() (app.serializers.role.RoleSerializer method), 293
 _normalize_nested_options() (app.serializers.user.UserExportWordSerializer method), 314
 _normalize_nested_options() (app.serializers.user.UserSerializer method), 321
 _parse_user_data() (in module app.celery.excel.tasks), 85
 _parser (app.blueprints.documents.DocumentResource attribute), 40
 _pk (app.models.base.Base property), 158
 _pk (app.models.document.Document property), 169
 _pk (app.models.role.Role property), 180
 _pk (app.models.user.User property), 196
 _pk (database.migrations.Migration property), 371
 _pk (database.migrations.aad_create_user_roles_table._OldUser property), 357
 _pk (database.migrations.aaf_remove_role_slug_column._OldRole property), 360
 _pk_expr() (app.models.base.Base method), 158
 _pk_expr() (app.models.document.Document method), 169
 _pk_expr() (app.models.role.Role method), 180
 _pk_expr() (app.models.user.User method), 196
 _pk_expr() (database.migrations.Migration method), 371
 _pk_expr() (database.migrations.aad_create_user_roles_table._OldUser method), 357
 _pk_expr() (database.migrations.aaf_remove_role_slug_column._OldRole method), 360
 _pool (app.celery.MyCelery attribute), 131
 _populate_unsaved_relations() (app.models.base.Base method), 158
 _populate_unsaved_relations() (app.models.document.Document method), 169
 _populate_unsaved_relations() (app.models.role.Role method), 180
 _populate_unsaved_relations() (app.models.user.User method), 196
 _populate_unsaved_relations() (database.migrations.Migration method), 371
 _populate_unsaved_relations() (database.migrations.aad_create_user_roles_table._OldUser method), 357
 _populate_unsaved_relations() (database.migrations.aaf_remove_role_slug_column._OldRole method), 360
 _prune_fields() (app.models.base.Base method), 158
 _prune_fields() (app.models.document.Document method), 169
 _prune_fields() (app.models.role.Role method), 180
 _prune_fields() (app.models.user.User method), 196
 _prune_fields() (database.migrations.Migration method), 371
 _prune_fields() (database.migrations.aad_create_user_roles_table._OldUser method), 357
 _prune_fields() (database.migrations.aaf_remove_role_slug_column._OldRole method), 360
 _register_blueprints() (in module app), 347
 _remove_test_files() (in module tests.conftest), 390
 _rename_celery_settings() (config.Meta class method), 432
 _rgetattr() (app.celery.MyCelery method), 131
 _run_validator() (app.serializers.auth.AuthUserConfirmResetPasswordSerializer method), 216

_run_validator() (app.serializers.auth.AuthUserLoginSerializer method), 222
 _run_validator() (app.serializers.auth.AuthUserLoginSerializer method), 222
 _run_validator() (app.serializers.core.SearchSerializer method), 237
 _run_validator() (app.serializers.core._SearchOrderSerializer method), 241
 _run_validator() (app.serializers.core._SearchOrderSerializer method), 245
 _run_validator() (app.serializers.core._SearchValueSerializer method), 251
 _run_validator() (app.serializers.core._SearchValueSerializer method), 251
 _run_validator() (app.serializers.document.DocumentAttachmentSerializer method), 269
 _run_validator() (app.serializers.document.DocumentAttachmentSerializer method), 269
 _run_validator() (app.serializers.document.DocumentSerializer property), 241
 _run_validator() (app.serializers.document.DocumentSerializer method), 276
 _run_validator() (app.serializers.role.RoleSerializer method), 293
 _run_validator() (app.serializers.role.RoleSerializer method), 293
 _run_validator() (app.serializers.user.UserExportWordSerializer method), 314
 _run_validator() (app.serializers.user.UserExportWordSerializer method), 314
 _run_validator() (app.serializers.user.UserSerializer method), 321
 _run_validator() (app.serializers.user.UserSerializer method), 321
 _schema (app.models.base.Base attribute), 158
 _schema (app.models.document.Document attribute), 169
 _schema (app.models.document.Document attribute), 169
 _schema (app.models.role.Role attribute), 180
 _schema (app.models.role.Role attribute), 180
 _schema (app.models.user.User attribute), 196
 _schema (app.models.user.User attribute), 196
 _schema (database.migrations.Migration attribute), 371
 _schema (database.migrations.aad_create_user_roles_table._OldUser attribute), 357
 _schema (database.migrations.aad_create_user_roles_table._OldUser attribute), 357
 _schema (database.migrations.aaf_remove_role_slug_column._OldRole attribute), 360
 _schema (database.migrations.aaf_remove_role_slug_column._OldRole attribute), 360
 _serialize() (app.serializers.auth.AuthUserConfirmResetSerializer method), 216
 _serialize() (app.serializers.auth.AuthUserConfirmResetSerializer method), 216
 _serialize() (app.serializers.auth.AuthUserLoginSerializer method), 222
 _serialize() (app.serializers.auth.AuthUserLoginSerializer method), 222
 _serialize() (app.serializers.core.SearchSerializer method), 237
 _serialize() (app.serializers.core.SearchSerializer method), 237
 _serialize() (app.serializers.core.TimestampField method), 241
 _serialize() (app.serializers.core.TimestampField method), 241
 _serialize() (app.serializers.core._SearchOrderSerializer method), 245
 _serialize() (app.serializers.core._SearchOrderSerializer method), 245
 _serialize() (app.serializers.core._SearchValueSerializer method), 251
 _serialize() (app.serializers.core._SearchValueSerializer method), 251
 _serialize() (app.serializers.document.DocumentAttachmentSerializer method), 269
 _serialize() (app.serializers.document.DocumentAttachmentSerializer method), 269
 _serialize() (app.serializers.document.DocumentSerializer method), 276
 _serialize() (app.serializers.document.DocumentSerializer method), 276
 _serialize() (app.serializers.role.RoleName method), 289
 _serialize() (app.serializers.role.RoleName method), 289
 _serialize() (app.serializers.role.RoleSerializer method), 293
 _serialize() (app.serializers.role.RoleSerializer method), 293
 _serialize() (app.serializers.user.UserExportWordSerializer method), 314
 _serialize() (app.serializers.user.UserExportWordSerializer method), 314
 _serialize() (app.serializers.user.UserSerializer method), 321
 _serialize() (app.serializers.user.UserSerializer method), 321
 _serialize() (app.serializers.user.VerifyRoleId method), 324
 _serialize() (app.serializers.user.VerifyRoleId method), 324
 _serialize_to_periodic_task_entry() (app.celery.MyCelery method), 131
 _serialize_to_periodic_task_entry() (app.celery.MyCelery method), 131
 _task_from_fun() (app.celery.MyCelery method), 131
 _task_from_fun() (app.celery.MyCelery method), 131
 _validate() (app.serializers.core.TimestampField method), 241
 _validate() (app.serializers.core.TimestampField method), 241
 _validate() (app.serializers.role.RoleName method), 289
 _validate() (app.serializers.role.RoleName method), 289
 _validate() (app.serializers.user.VerifyRoleId method), 325
 _validate() (app.serializers.user.VerifyRoleId method), 325
 _validate_all (app.serializers.core.TimestampField method), 241
 _validate_all (app.serializers.core.TimestampField method), 241
 _validate_all (app.serializers.role.RoleName property), 289
 _validate_all (app.serializers.role.RoleName property), 289
 _validate_all (app.serializers.user.VerifyRoleId property), 325
 _validate_all (app.serializers.user.VerifyRoleId property), 325
 _validate_missing() (app.serializers.core.TimestampField method), 241
 _validate_missing() (app.serializers.core.TimestampField method), 241
 _validate_missing() (app.serializers.role.RoleName method), 289
 _validate_missing() (app.serializers.role.RoleName method), 289
 _validate_missing() (app.serializers.user.VerifyRoleId method), 325
 _validate_missing() (app.serializers.user.VerifyRoleId method), 325
 abstract (app.celery.ContextTask attribute), 89, 121
 abstract (app.celery.ContextTask attribute), 89, 121
 accept_content (config.Config attribute), 405, 430
 accept_content (config.Config attribute), 405, 430
 accept_content (config.DevConfig attribute), 412, 432
 accept_content (config.DevConfig attribute), 412, 432
 accept_content (config.ProdConfig attribute), 420, 434
 accept_content (config.ProdConfig attribute), 420, 434
 accept_content (config.TestConfig attribute), 428, 435
 accept_content (config.TestConfig attribute), 428, 435
 acks_late (app.celery.ContextTask attribute), 89, 121
 acks_late (app.celery.ContextTask attribute), 89, 121
 acks_on_failure_or_timeout (app.celery.ContextTask attribute), 89, 121
 acks_on_failure_or_timeout (app.celery.ContextTask attribute), 89, 121
 active (app.models.user.User attribute), 184, 196
 active (app.models.user.User attribute), 184, 196
 active (database.migrations.aad_create_user_roles_table._OldUser attribute), 357
 active (database.migrations.aad_create_user_roles_table._OldUser attribute), 357
 add_around() (app.celery.ContextTask class method), 94, 121
 add_around() (app.celery.ContextTask class method), 94, 121
 add_defaults() (app.celery.MyCelery method), 111, 131
 add_defaults() (app.celery.MyCelery method), 111, 131
 add_index() (app.models.base.Base class method), 153, 158
 add_index() (app.models.base.Base class method), 153, 158
 add_index() (app.models.document.Document class method), 164, 169
 add_index() (app.models.document.Document class method), 164, 169
 add_index() (app.models.role.Role class method), 175, 180
 add_index() (app.models.role.Role class method), 175, 180
 add_index() (app.models.user.User class method), 188, 196
 add_index() (app.models.user.User class method), 188, 196
 add_index() (database.migrations.aad_create_user_roles_table._OldUser class method), 357
 add_index() (database.migrations.aad_create_user_roles_table._OldUser class method), 357
 add_index() (database.migrations.aaf_remove_role_slug_column._OldRole class method), 360
 add_index() (database.migrations.aaf_remove_role_slug_column._OldRole class method), 360
 add_index() (database.migrations.Migration class method), 366, 371
 add_index() (database.migrations.Migration class method), 366, 371

[add_periodic_task\(\)](#) (*app.celery.MyCelery* method), 111, 131
[add_permissions\(\)](#) (*app.models.role.Role* method), 175, 180
[add_permissions\(\)](#) (*database.migrations.aaf_remove_role_slug_column._OldRole* method), 360
[add_to_chord\(\)](#) (*app.celery.ContextTask* method), 94, 121
[add_trail\(\)](#) (*app.celery.ContextTask* method), 95, 121
[AddCreatedByColumnOnUserTable](#) (class in *database.migrations.aab_add_created_by_column_on_user_table*), 354
[AddFsUniquifierColumnOnUsersTable](#) (class in *database.migrations.aag_add_fs_uniquifier_column_on_users_table*), 362, 363
[AddGenreColumnOnUserTable](#) (class in *database.migrations.aaa_add_genre_column_on_user_table*), 353
[after_request\(\)](#) (*tests.custom_flask_client.CustomFlaskClient* static method), 392, 395
[after_return\(\)](#) (*app.celery.ContextTask* method), 95, 121
[alias\(\)](#) (*app.models.base.Base* class method), 154, 158
[alias\(\)](#) (*app.models.document.Document* class method), 164, 169
[alias\(\)](#) (*app.models.role.Role* class method), 175, 180
[alias\(\)](#) (*app.models.user.User* class method), 188, 196
[alias\(\)](#) (*database.migrations.aad_create_user_roles_table._OldRole* class method), 357
[alias\(\)](#) (*database.migrations.aaf_remove_role_slug_column._OldRole* class method), 360
[alias\(\)](#) (*database.migrations.Migration* class method), 366, 371
[ALLOWED_CONTENT_TYPES](#) (*config.Config* attribute), 401, 429
[ALLOWED_CONTENT_TYPES](#) (*config.DevConfig* attribute), 409, 431
[ALLOWED_CONTENT_TYPES](#) (*config.ProdConfig* attribute), 417, 433
[ALLOWED_CONTENT_TYPES](#) (*config.TestConfig* attribute), 424, 434
[ALLOWED_MIME_TYPES](#) (*config.Config* attribute), 401, 429
[ALLOWED_MIME_TYPES](#) (*config.DevConfig* attribute), 409, 431
[ALLOWED_MIME_TYPES](#) (*config.ProdConfig* attribute), 417, 433
[ALLOWED_MIME_TYPES](#) (*config.TestConfig* attribute), 424, 434
[amqp](#) (*app.celery.MyCelery* attribute), 105, 131
[amqp_cls](#) (*app.celery.MyCelery* attribute), 106, 131
[annotate\(\)](#) (*app.celery.ContextTask* class method), 95, 122
[annotations](#) (*app.celery.MyCelery* attribute), 106, 131
[app](#)
 module, 8
[app](#) (*app.celery.ContextTask* attribute), 89, 122
[app\(\)](#) (in module *tests.conftest*), 390
[app.blueprints](#)
 module, 8
 [app.blueprints.auth](#)
 module, 9
 [app.blueprints.base](#)
 module, 23
 [app.blueprints.documents](#)
 module, 29
 [app.blueprints.roles](#)
 module, 46
 [app.blueprints.tasks](#)
 module, 55
 [app.blueprints.users](#)
 module, 61
 [app.celery](#)
 module, 84
 [app.celery.excel](#)
 module, 85
 [app.celery.excel.tasks](#)
 module, 85
 [app.celery.tasks](#)
 module, 86
 [app.celery.tests](#)
 module, 86
 [app.celery.tests.tasks](#)
 module, 86
 [app.celery.word](#)
 module, 87
 [app.celery.word.tasks](#)
 module, 87
 [app.exceptions](#)
 module, 139
 [app.extensions](#)
 module, 140
 [app.managers](#)
 module, 141
 [app.managers.base](#)
 module, 141
 [app.managers.document](#)
 module, 143
 [app.managers.role](#)
 module, 145
 [app.managers.user](#)
 module, 146
 [app.middleware](#)
 module, 149
 [app.models](#)
 module, 150
 [app.models.base](#)
 module, 151

`app.models.document`
 module, 160

`app.models.role`
 module, 171

`app.models.user`
 module, 182

`app.models.user_roles`
 module, 200

`app.serializers`
 module, 200

`app.serializers.auth`
 module, 201

`app.serializers.core`
 module, 225

`app.serializers.document`
 module, 254

`app.serializers.role`
 module, 279

`app.serializers.user`
 module, 297

`app.services`
 module, 326

`app.services.auth`
 module, 327

`app.services.base`
 module, 328

`app.services.document`
 module, 330

`app.services.role`
 module, 331

`app.services.task`
 module, 333

`app.services.user`
 module, 335

`app.swagger`
 module, 336

`app.swagger.auth`
 module, 337

`app.swagger.core`
 module, 337

`app.swagger.document`
 module, 337

`app.swagger.role`
 module, 337

`app.swagger.user`
 module, 337

`app.utils`
 module, 337

`app.utils.constants`
 module, 337

`app.utils.decorators`
 module, 338

`app.utils.file_storage`
 module, 338

`app.utils.libreoffice`
 module, 340

`app.utils.request_query_operator`
 module, 341

`application` (*tests.custom_flask_client.CustomFlaskClient* attribute), 395

`apply()` (*app.celery.ContextTask* method), 95, 122

`apply_async()` (*app.celery.ContextTask* method), 95, 122

`args` (*app.celery.ContextTask.MaxRetriesExceededError* attribute), 120

`args` (*app.celery.ContextTask.OperationalError* attribute), 120

`args` (*app.exceptions.FileEmptyError* attribute), 140

`args` (*app.utils.libreoffice.LibreOfficeError* attribute), 341

`as_view()` (*app.blueprints.auth.AuthBaseResource* class method), 11, 20

`as_view()` (*app.blueprints.auth.AuthUserLoginResource* class method), 13, 21

`as_view()` (*app.blueprints.auth.AuthUserLogoutResource* class method), 15, 21

`as_view()` (*app.blueprints.auth.RequestResetPasswordResource* class method), 17, 22

`as_view()` (*app.blueprints.auth.ResetPasswordResource* class method), 20, 23

`as_view()` (*app.blueprints.base.BaseResource* class method), 25, 28

`as_view()` (*app.blueprints.base.WelcomeResource* class method), 27, 28

`as_view()` (*app.blueprints.documents.DocumentBaseResource* class method), 31, 39

`as_view()` (*app.blueprints.documents.DocumentResource* class method), 34, 40

`as_view()` (*app.blueprints.documents.NewDocumentResource* class method), 36, 40

`as_view()` (*app.blueprints.documents.SearchDocumentResource* class method), 39, 41

`as_view()` (*app.blueprints.roles.NewRoleResource* class method), 44, 52

`as_view()` (*app.blueprints.roles.RoleBaseResource* class method), 47, 53

`as_view()` (*app.blueprints.roles.RoleResource* class method), 49, 53

`as_view()` (*app.blueprints.roles.RolesSearchResource* class method), 52, 54

`as_view()` (*app.blueprints.tasks.TaskResource* class method), 57, 59

`as_view()` (*app.blueprints.tasks.TaskStatusResource* class method), 59, 60

`as_view()` (*app.blueprints.users.ExportUsersExcelAndWordResource* class method), 63, 79

`as_view()` (*app.blueprints.users.ExportUsersExcelResource* class method), 66, 80

as_view() (*app.blueprints.users.ExportUsersWordResource* class method), 69, 80
 as_view() (*app.blueprints.users.NewUserResource* class method), 71, 81
 as_view() (*app.blueprints.users.UserBaseResource* class method), 74, 82
 as_view() (*app.blueprints.users.UserResource* class method), 76, 83
 as_view() (*app.blueprints.users.UsersSearchResource* class method), 79, 83
 AsyncResult (*app.celery.MyCelery* attribute), 104, 129
 AsyncResult() (*app.celery.ContextTask* method), 94, 120
 auth_header() (in module *tests.conftest*), 390
 auth_service (*app.blueprints.auth.AuthBaseResource* attribute), 9, 20
 auth_service (*app.blueprints.auth.AuthUserLoginResource* attribute), 12, 21
 auth_service (*app.blueprints.auth.AuthUserLogoutResource* attribute), 14, 21
 auth_service (*app.blueprints.auth.RequestResetPasswordResource* attribute), 16, 22
 auth_service (*app.blueprints.auth.ResetPasswordResource* attribute), 18, 23
 AuthBaseResource (class in *app.blueprints.auth*), 9, 20
 AuthService (class in *app.services.auth*), 327, 328
 AuthUserConfirmResetPasswordSerializer (class in *app.serializers.auth*), 201, 213
 AuthUserConfirmResetPasswordSerializer.Meta (class in *app.serializers.auth*), 213
 AuthUserLoginResource (class in *app.blueprints.auth*), 11, 21
 AuthUserLoginSerializer (class in *app.serializers.auth*), 207, 219
 AuthUserLoginSerializer.Meta (class in *app.serializers.auth*), 219
 AuthUserLogoutResource (class in *app.blueprints.auth*), 13, 21
 autodiscover_tasks() (*app.celery.MyCelery* method), 111, 131

B

backend (*app.celery.ContextTask* property), 89, 123
 backend (*app.celery.MyCelery* property), 106, 132
 backend_cls (*app.celery.MyCelery* attribute), 106, 132
 Base (class in *app.models.base*), 151, 158
 BaseManager (class in *app.managers.base*), 141, 143
 BaseResource (class in *app.blueprints.base*), 24, 28
 BaseService (class in *app.services.base*), 328, 329
 Beat (*app.celery.MyCelery* attribute), 104, 129
 before_request() (*tests.custom_flask_client.CustomFlaskClient* static method), 392, 395
 before_start() (*app.celery.ContextTask* method), 97, 123
 bind() (*app.celery.ContextTask* class method), 97, 124
 bind() (*app.models.base.Base* class method), 154, 158
 bind() (*app.models.document.Document* class method), 164, 169
 bind() (*app.models.role.Role* class method), 175, 180
 bind() (*app.models.user.User* class method), 189, 196
 bind() (*database.migrations.aad_create_user_roles_table._OldUser* class method), 357
 bind() (*database.migrations.aaf_remove_role_slug_column._OldRole* class method), 360
 bind() (*database.migrations.Migration* class method), 366, 371
 bind_ctx() (*app.models.base.Base* class method), 154, 158
 bind_ctx() (*app.models.document.Document* class method), 164, 169
 bind_ctx() (*app.models.role.Role* class method), 175, 180
 bind_ctx() (*app.models.user.User* class method), 189, 196
 bind_ctx() (*database.migrations.aad_create_user_roles_table._OldUser* class method), 357
 bind_ctx() (*database.migrations.aaf_remove_role_slug_column._OldRole* class method), 360
 bind_ctx() (*database.migrations.Migration* class method), 366, 371
 birth_date (*app.models.user.User* attribute), 184, 196
 birth_date (*database.migrations.aad_create_user_roles_table._OldUser* attribute), 357
 broker_connection() (*app.celery.MyCelery* method), 112, 132
 broker_url (*config.Config* attribute), 405, 430
 broker_url (*config.DevConfig* attribute), 412, 432
 broker_url (*config.ProdConfig* attribute), 420, 434
 broker_url (*config.TestConfig* attribute), 428, 435
 bugreport() (*app.celery.MyCelery* method), 113, 132
 build_clause_operators() (*app.utils.request_query_operator.Helper* method), 342, 343
 build_order_by() (*app.utils.request_query_operator.Helper* static method), 342, 343
 build_sql_expression() (*app.utils.request_query_operator.Helper* method), 342, 344
 build_string_clause() (*app.utils.request_query_operator.Helper* method), 342, 344
 builtin_fixups (*app.celery.MyCelery* attribute), 106, 132
 bulk_create() (*app.models.base.Base* class method), 154, 158
 bulk_create() (*app.models.document.Document* class method), 164, 169
 bulk_create() (*app.models.role.Role* class method),

- 175, 180
- `bulk_create()` (*app.models.user.User* class method), 189, 196
- `bulk_create()` (*database.migrations.aad_create_user_roles_table._OldUser* class method), 357
- `bulk_create()` (*database.migrations.aaf_remove_role_slug_column._OldRole* class method), 360
- `bulk_create()` (*database.migrations.Migration* class method), 366, 371
- `bulk_update()` (*app.models.base.Base* class method), 154, 158
- `bulk_update()` (*app.models.document.Document* class method), 164, 169
- `bulk_update()` (*app.models.role.Role* class method), 175, 180
- `bulk_update()` (*app.models.user.User* class method), 189, 196
- `bulk_update()` (*database.migrations.aad_create_user_roles_table._OldUser* class method), 357
- `bulk_update()` (*database.migrations.aaf_remove_role_slug_column._OldRole* class method), 360
- `bulk_update()` (*database.migrations.Migration* class method), 366, 371
- C**
- `calc_username()` (*app.models.user.User* method), 189, 196
- `characters_written` (*app.exceptions.FileEmptyError* attribute), 140
- `check_task_status()` (*app.services.task.TaskService* method), 334
- `check_token_status()` (*app.services.auth.AuthService* method), 327, 328
- `children` (*app.models.user.User* attribute), 184, 196
- `children` (*database.migrations.aad_create_user_roles_table._OldUser* attribute), 357
- `chunks()` (*app.celery.ContextTask* method), 97, 124
- `client()` (in module *tests.conftest*), 390
- `clone()` (*app.models.base.Base* method), 154, 158
- `clone()` (*app.models.document.Document* method), 165, 169
- `clone()` (*app.models.role.Role* method), 176, 180
- `clone()` (*app.models.user.User* method), 189, 196
- `clone()` (*database.migrations.aad_create_user_roles_table._OldUser* method), 357
- `clone()` (*database.migrations.aaf_remove_role_slug_column._OldRole* method), 360
- `clone()` (*database.migrations.Migration* method), 366, 371
- `close()` (*app.celery.MyCelery* method), 113, 132
- `coerce()` (*app.models.base.Base* method), 154, 158
- `coerce()` (*app.models.document.Document* method), 165, 169
- `coerce()` (*app.models.role.Role* method), 176, 180
- `coerce()` (*app.models.user.User* method), 189, 196
- `coerce()` (*database.migrations.aad_create_user_roles_table._OldUser* method), 357
- `coerce()` (*database.migrations.aaf_remove_role_slug_column._OldRole* method), 360
- `coerce()` (*database.migrations.Migration* method), 366, 371
- `conf` (*app.celery.MyCelery* property), 106, 133
- `config` module, 399
- `Config` (class in *config*), 399, 429
- `config_from_cmdline()` (*app.celery.MyCelery* method), 113, 133
- `config_from_envvar()` (*app.celery.MyCelery* method), 113, 133
- `config_from_object()` (*app.celery.MyCelery* method), 113, 133
- `confirm_request_reset_password()` (*app.services.auth.AuthService* method), 327, 328
- `connection()` (*app.celery.MyCelery* method), 114, 133
- `connection_for_read()` (*app.celery.MyCelery* method), 114, 134
- `connection_for_write()` (*app.celery.MyCelery* method), 115, 134
- `connection_or_acquire()` (*app.celery.MyCelery* method), 115, 134
- `context` (*app.serializers.core.TimestampField* property), 231, 241
- `context` (*app.serializers.role.RoleName* property), 280, 289
- `context` (*app.serializers.user.VerifyRoleId* property), 309, 325
- `ContextTask` (class in *app.celery*), 87, 120
- `ContextTask.MaxRetriesExceededError`, 120
- `ContextTask.OperationalError`, 120
- `control` (*app.celery.MyCelery* attribute), 106, 134
- `control_cls` (*app.celery.MyCelery* attribute), 106, 134
- `convert_to()` (in module *app.utils.libreoffice*), 340, 341
- `copy()` (*app.models.base.Base* static method), 154, 158
- `copy()` (*app.models.document.Document* static method), 165, 169
- `copy()` (*app.models.role.Role* static method), 176, 180
- `copy()` (*app.models.user.User* static method), 189, 196
- `copy()` (*database.migrations.aad_create_user_roles_table._OldUser* static method), 357
- `copy()` (*database.migrations.aaf_remove_role_slug_column._OldRole* static method), 360
- `copy()` (*database.migrations.Migration* static method), 367, 371
- `copy_file()` (*app.utils.file_storage.FileStorage* static method), 339, 340
- `create()` (*app.managers.base.BaseManager* method),

[142, 143](#)
[create\(\)](#) (*app.managers.document.DocumentManager* method), [144](#)
[create\(\)](#) (*app.managers.role.RoleManager* method), [146](#)
[create\(\)](#) (*app.managers.user.UserManager* method), [147, 148](#)
[create\(\)](#) (*app.models.base.Base* class method), [154, 158](#)
[create\(\)](#) (*app.models.document.Document* class method), [165, 169](#)
[create\(\)](#) (*app.models.role.Role* class method), [176, 180](#)
[create\(\)](#) (*app.models.user.User* class method), [190, 196](#)
[create\(\)](#) (*app.services.base.BaseService* method), [329](#)
[create\(\)](#) (*app.services.document.DocumentService* method), [331](#)
[create\(\)](#) (*app.services.role.RoleService* method), [332, 333](#)
[create\(\)](#) (*app.services.user.UserService* method), [335, 336](#)
[create\(\)](#) (*database.migrations.aad_create_user_roles_table* class method), [357](#)
[create\(\)](#) (*database.migrations.aaf_remove_role_slug_column._OldRole* class method), [360](#)
[create\(\)](#) (*database.migrations.Migration* class method), [367, 371](#)
[create_app\(\)](#) (in module *app*), [347](#)
[create_search_query\(\)](#) (*app.utils.request_query_operator.RequestQueryOperator* tests.custom_flask_client), [391, 395](#) static method), [343, 344](#)
[create_table\(\)](#) (*app.models.base.Base* class method), [155, 159](#)
[create_table\(\)](#) (*app.models.document.Document* class method), [165, 169](#)
[create_table\(\)](#) (*app.models.role.Role* class method), [176, 180](#)
[create_table\(\)](#) (*app.models.user.User* class method), [190, 196](#)
[create_table\(\)](#) (*database.migrations.aad_create_user_roles_table._OldUser* class method), [357](#)
[create_table\(\)](#) (*database.migrations.aaf_remove_role_slug_column._OldRole* class method), [360](#)
[create_table\(\)](#) (*database.migrations.Migration* class method), [367, 371](#)
[create_task_cls\(\)](#) (*app.celery.MyCelery* method), [115, 134](#)
[create_task_table\(\)](#) (in module *app.celery.tests.tasks*), [86](#)
[create_user_email_task\(\)](#) (in module *app.celery.tasks*), [86](#)
[create_word_and_excel_documents_task\(\)](#) (in module *app.celery.tasks*), [86](#)
[created_at](#) (*app.models.document.Document* attribute), [161, 169](#)
[created_at](#) (*app.models.role.Role* attribute), [172, 180](#)
[created_at](#) (*app.models.user.User* attribute), [184, 196](#)
[created_at](#) (*database.migrations.aad_create_user_roles_table._OldUser* attribute), [357](#)
[created_at](#) (*database.migrations.aaf_remove_role_slug_column._OldRole* attribute), [360](#)
[created_by](#) (*app.models.document.Document* attribute), [161, 169](#)
[created_by](#) (*app.models.user.User* attribute), [184, 197](#)
[created_by](#) (*database.migrations.aad_create_user_roles_table._OldUser* attribute), [357](#)
[created_by_id](#) (*app.models.document.Document* attribute), [161, 169](#)
[created_by_id](#) (*app.models.user.User* attribute), [184, 197](#)
[created_by_id](#) (*database.migrations.aad_create_user_roles_table._OldUser* attribute), [357](#)
[CreateDocumentsTable](#) (class in *database.migrations.aac_create_documents_table*), [355](#)
[CreateUserRolesTable](#) (class in *database.migrations.aad_create_user_roles_table*), [356](#)
[current_task](#) (*app.celery.MyCelery* property), [106, 134](#)
[current_worker_task](#) (*app.celery.MyCelery* property), [107, 134](#)
[CustomFlaskClient](#) (class in *tests.custom_flask_client*), [391, 395](#)

D

[database](#) module, [348](#)
[DATABASE](#) (*config.Config* attribute), [402, 429](#)
[DATABASE](#) (*config.DevConfig* attribute), [409, 431](#)
[DATABASE](#) (*config.ProdConfig* attribute), [417, 433](#)
[DATABASE](#) (*config.TestConfig* attribute), [424, 434](#)
[database.factories](#) module, [348](#)
[database.migrations](#) module, [352](#)
[database.migrations.aaa_add_genre_column_on_user_table](#) module, [353](#)
[database.migrations.aab_add_created_by_column_on_user_table](#) module, [354](#)
[database.migrations.aac_create_documents_table](#) module, [355](#)
[database.migrations.aad_create_user_roles_table](#) module, [356](#)
[database.migrations.aaf_remove_role_slug_column](#) module, [359](#)
[database.migrations.aag_add_fs_uniquifier_column_on_users](#) module, [362](#)
[database.seeds](#)

- module, 373
- database.seeds.document_seeder
 - module, 373
- database.seeds.role_seeder
 - module, 374
- database.seeds.user_seeder
 - module, 375
- DEBUG (*config.Config* attribute), 402, 429
- DEBUG (*config.DevConfig* attribute), 409, 431
- DEBUG (*config.ProdConfig* attribute), 417, 433
- DEBUG (*config.TestConfig* attribute), 425, 434
- decorators (*app.blueprints.auth.AuthBaseResource* attribute), 10, 20
- decorators (*app.blueprints.auth.AuthUserLoginResource* attribute), 12, 21
- decorators (*app.blueprints.auth.AuthUserLogoutResource* attribute), 14, 21
- decorators (*app.blueprints.auth.RequestResetPasswordResource* attribute), 16, 22
- decorators (*app.blueprints.auth.ResetPasswordResource* attribute), 18, 23
- decorators (*app.blueprints.base.BaseResource* attribute), 24, 28
- decorators (*app.blueprints.base.WelcomeResource* attribute), 26, 28
- decorators (*app.blueprints.documents.DocumentBaseResource* attribute), 30, 39
- decorators (*app.blueprints.documents.DocumentResource* attribute), 32, 40
- decorators (*app.blueprints.documents.NewDocumentResource* attribute), 35, 41
- decorators (*app.blueprints.documents.SearchDocumentResource* attribute), 37, 41
- decorators (*app.blueprints.roles.NewRoleResource* attribute), 43, 52
- decorators (*app.blueprints.roles.RoleBaseResource* attribute), 45, 53
- decorators (*app.blueprints.roles.RoleResource* attribute), 48, 54
- decorators (*app.blueprints.roles.RolesSearchResource* attribute), 50, 54
- decorators (*app.blueprints.tasks.TaskResource* attribute), 56, 59
- decorators (*app.blueprints.tasks.TaskStatusResource* attribute), 58, 60
- decorators (*app.blueprints.users.ExportUsersExcelAndWordResource* attribute), 62, 79
- decorators (*app.blueprints.users.ExportUsersExcelResource* attribute), 65, 80
- decorators (*app.blueprints.users.ExportUsersWordResource* attribute), 67, 81
- decorators (*app.blueprints.users.NewUserResource* attribute), 70, 81
- decorators (*app.blueprints.users.UserBaseResource* attribute), 72, 82
- decorators (*app.blueprints.users.UserResource* attribute), 75, 83
- decorators (*app.blueprints.users.UsersSearchResource* attribute), 77, 83
- default (*app.serializers.core.TimestampField* property), 232, 241
- default (*app.serializers.role.RoleName* property), 280, 289
- default (*app.serializers.user.VerifyRoleId* property), 309, 325
- default_connection() (*app.celery.MyCelery* method), 115, 134
- default_error_messages (*app.serializers.core.TimestampField* attribute), 232, 241
- default_error_messages (*app.serializers.role.RoleName* attribute), 280, 289
- default_error_messages (*app.serializers.user.VerifyRoleId* attribute), 309, 325
- default_producer() (*app.celery.MyCelery* method), 115, 135
- default_retry_delay (*app.celery.ContextTask* attribute), 90, 124
- delay() (*app.celery.ContextTask* method), 97, 124
- delete() (*app.blueprints.documents.DocumentResource* method), 34, 40
- delete() (*app.blueprints.roles.RoleResource* method), 49, 54
- delete() (*app.blueprints.users.UserResource* method), 76, 83
- delete() (*app.managers.base.BaseManager* method), 142, 143
- delete() (*app.managers.document.DocumentManager* method), 144
- delete() (*app.managers.role.RoleManager* method), 146
- delete() (*app.managers.user.UserManager* method), 148
- delete() (*app.models.base.Base* class method), 155, 159
- delete() (*app.models.document.Document* class method), 165, 169
- delete() (*app.models.role.Role* class method), 176, 181
- delete() (*app.models.user.User* class method), 190, 197
- delete() (*app.services.base.BaseService* method), 329
- delete() (*app.services.document.DocumentService* method), 331
- delete() (*app.services.role.RoleService* method), 332, 333
- delete() (*app.services.user.UserService* method), 335, 336

`delete()` (`database.migrations.aad_create_user_roles_table._OldUserRoles` class method), 357
`delete()` (`database.migrations.aaf_remove_role_slug_column._OldRole` class method), 360
`delete()` (`database.migrations.Migration` class method), 367, 371
`delete()` (`tests.custom_flask_client.CustomFlaskClient` method), 392, 395
`delete_by_id()` (`app.models.base.Base` class method), 155, 159
`delete_by_id()` (`app.models.document.Document` class method), 165, 169
`delete_by_id()` (`app.models.role.Role` class method), 176, 181
`delete_by_id()` (`app.models.user.User` class method), 190, 197
`delete_by_id()` (`database.migrations.aad_create_user_roles_table._OldUserRoles` class method), 357
`delete_by_id()` (`database.migrations.aaf_remove_role_slug_column._OldRole` class method), 360
`delete_by_id()` (`database.migrations.Migration` class method), 367, 371
`delete_cookie()` (`tests.custom_flask_client.CustomFlaskClient` method), 392, 395
`delete_instance()` (`app.models.base.Base` method), 155, 159
`delete_instance()` (`app.models.document.Document` method), 165, 169
`delete_instance()` (`app.models.role.Role` method), 176, 181
`delete_instance()` (`app.models.user.User` method), 190, 197
`delete_instance()` (`database.migrations.aad_create_user_roles_table._OldUserRoles` class method), 357
`delete_instance()` (`database.migrations.aaf_remove_role_slug_column._OldRole` class method), 360
`delete_instance()` (`database.migrations.Migration` method), 367, 371
`deleted_at` (`app.models.document.Document` attribute), 161, 169
`deleted_at` (`app.models.role.Role` attribute), 172, 181
`deleted_at` (`app.models.user.User` attribute), 184, 197
`deleted_at` (`database.migrations.aad_create_user_roles_table._OldUserRoles` attribute), 357
`deleted_at` (`database.migrations.aaf_remove_role_slug_column._OldRole` attribute), 360
`dependencies()` (`app.models.base.Base` method), 155, 159
`dependencies()` (`app.models.document.Document` method), 165, 169
`dependencies()` (`app.models.role.Role` method), 176, 181
`dependencies()` (`app.models.user.User` method), 190, 197
`dependencies()` (`database.migrations.aad_create_user_roles_table._OldUserRoles` method), 357
`dependencies()` (`database.migrations.aaf_remove_role_slug_column._OldRole` method), 360
`dependencies()` (`database.migrations.Migration` method), 367, 371
`description` (`app.models.role.Role` attribute), 172, 181
`description` (`database.migrations.aaf_remove_role_slug_column._OldRole` attribute), 360
`deserialize()` (`app.serializers.core.TimestampField` method), 233, 241
`deserialize()` (`app.serializers.role.RoleName` method), 281, 289
`deserialize()` (`app.serializers.user.VerifyRoleId` method), 310, 325
`DevConfig` (class in `config`), 407, 431
`DEVELOPMENT` (`config.DevConfig` attribute), 402, 429
`DEVELOPMENT` (`config.DevConfig` attribute), 409, 431
`DEVELOPMENT` (`config.ProdConfig` attribute), 417, 433
`DEVELOPMENT` (`config.TestConfig` attribute), 425, 434
`dict_class` (`app.serializers.auth.AuthUserConfirmResetPasswordSerializer` property), 202, 216
`dict_class` (`app.serializers.auth.AuthUserLoginSerializer` property), 208, 222
`dict_class` (`app.serializers.core._SearchOrderSerializer` property), 245
`dict_class` (`app.serializers.core._SearchValueSerializer` property), 251
`dict_class` (`app.serializers.core.SearchSerializer` property), 226, 237
`dict_class` (`app.serializers.document.DocumentAttachmentSerializer` property), 256, 269
`dict_class` (`app.serializers.document.DocumentSerializer` property), 261, 276
`dict_class` (`app.serializers.role.RoleSerializer` property), 283, 294
`dict_class` (`app.serializers.user.UserExportWordSerializer` property), 298, 314
`dict_class` (`app.serializers.user.UserSerializer` property), 304, 321
`directory_path` (`app.models.document.Document` attribute), 161, 169
`dirty_fields` (`app.models.base.Base` property), 151, 159
`dirty_fields` (`app.models.document.Document` property), 161, 169
`dirty_fields` (`app.models.role.Role` property), 172, 181
`dirty_fields` (`app.models.user.User` property), 184, 197
`dirty_fields` (`database.migrations.aad_create_user_roles_table._OldUserRoles` property), 357
`dirty_fields` (`database.migrations.aaf_remove_role_slug_column._OldRole` property), 360

dirty_fields (*database.migrations.Migration* property), 363, 371

dispatch_request() (*app.blueprints.auth.AuthBaseResource* method), 11, 20

dispatch_request() (*app.blueprints.auth.AuthUserLoginResource* method), 13, 21

dispatch_request() (*app.blueprints.auth.AuthUserLogoutResource* method), 15, 22

dispatch_request() (*app.blueprints.auth.RequestResetPasswordResource* method), 17, 22

dispatch_request() (*app.blueprints.auth.ResetPasswordResource* method), 20, 23

dispatch_request() (*app.blueprints.base.BaseResource* method), 25, 28

dispatch_request() (*app.blueprints.base.WelcomeResource* method), 27, 28

dispatch_request() (*app.blueprints.documents.DocumentBaseResource* method), 31, 39

dispatch_request() (*app.blueprints.documents.DocumentResource* method), 34, 40

dispatch_request() (*app.blueprints.documents.NewDocumentResource* method), 37, 41

dispatch_request() (*app.blueprints.documents.SearchDocumentResource* method), 39, 41

dispatch_request() (*app.blueprints.roles.NewRoleResource* method), 44, 52

dispatch_request() (*app.blueprints.roles.RoleBaseResource* method), 47, 53

dispatch_request() (*app.blueprints.roles.RoleResource* method), 49, 54

dispatch_request() (*app.blueprints.roles.RolesSearchResource* method), 52, 54

dispatch_request() (*app.blueprints.tasks.TaskResource* method), 57, 59

dispatch_request() (*app.blueprints.tasks.TaskStatusResource* method), 59, 60

dispatch_request() (*app.blueprints.users.ExportUsersEndpointResource* method), 64, 79

dispatch_request() (*app.blueprints.users.ExportUsersEndpointResource* method), 66, 80

dispatch_request() (*app.blueprints.users.ExportUsersEndpointResource* method), 69, 81

dispatch_request() (*app.blueprints.users.NewUserResource* method), 71, 81

dispatch_request() (*app.blueprints.users.UserBaseResource* method), 74, 82

dispatch_request() (*app.blueprints.users.UserResource* method), 76, 83

dispatch_request() (*app.blueprints.users.UsersSearchResource* method), 79, 84

doc_serializer (*app.blueprints.documents.DocumentBaseResource* attribute), 30, 39

doc_serializer (*app.blueprints.documents.DocumentResource* attribute), 32, 40

doc_serializer (*app.blueprints.documents.NewDocumentResource* attribute), 35, 41

doc_serializer (*app.blueprints.documents.SearchDocumentResource* attribute), 38, 42

Document (*class in app.models.document*), 160, 169

document_set (*app.models.user.User* attribute), 184, 197

DocumentAttachmentSerializer (*class in app.serializers.document*), 255, 266

DocumentAttachmentSerializer.Meta (*class in app.serializers.document*), 266

DocumentBaseResource (*class in app.blueprints.documents*), 29, 39

DocumentManager (*class in app.managers.document*), 43, 44

DocumentResource (*class in app.blueprints.documents*), 32, 40

DocumentSeeder (*class in database.seeds.document_seeder*), 373, 374

DocumentSerializer (*class in app.serializers.document*), 260, 272

DocumentSerializer.Meta (*class in app.serializers.document*), 272

DocumentService (*class in app.services.document*), 330, 331

DoesNotExist (*app.models.base.Base* attribute), 158

DoesNotExist (*app.models.document.Document* attribute), 169

DoesNotExist (*app.models.role.Role* attribute), 180

DoesNotExist (*app.models.user.User* attribute), 196

DoesNotExist (*database.migrations.aad_create_user_roles_table._OldUserRoles* attribute), 357

DoesNotExist (*database.migrations.aaf_remove_role_slug_column._OldRoleSlugs* attribute), 360

DoesNotExist (*database.migrations.Migration* attribute), 371

down() (*database.migrations.aaa_add_genre_column_on_user_table.AddGenreColumn* method), 353

down() (*database.migrations.aab_add_created_by_column_on_user_table.AddCreatedByColumn* method), 354

down() (*database.migrations.aac_create_documents_table.CreateDocumentsTable* method), 355

down() (*database.migrations.aad_create_user_roles_table.CreateUserRolesTable* method), 356

down() (*database.migrations.aaf_remove_role_slug_column.RemoveRoleSlugsTable* method), 359, 360

`down()` (`database.migrations.aag_add_fs_uniquifier_column` method), 363

`drop_table()` (`app.models.base.Base` class method), 155, 159

`drop_table()` (`app.models.document.Document` class method), 166, 169

`drop_table()` (`app.models.role.Role` class method), 177, 181

`drop_table()` (`app.models.user.User` class method), 190, 197

`drop_table()` (`database.migrations.aad_create_user_roles_table` class method), 357

`drop_table()` (`database.migrations.aaf_remove_role_slug_column` class method), 361

`drop_table()` (`database.migrations.Migration` class method), 367, 371

`dump()` (`app.serializers.auth.AuthUserConfirmResetPasswordSerializer` method), 203, 216

`dump()` (`app.serializers.auth.AuthUserLoginSerializer` method), 209, 222

`dump()` (`app.serializers.core._SearchOrderSerializer` method), 245

`dump()` (`app.serializers.core._SearchValueSerializer` method), 251

`dump()` (`app.serializers.core.SearchSerializer` method), 227, 237

`dump()` (`app.serializers.document.DocumentAttachmentSerializer` method), 257, 269

`dump()` (`app.serializers.document.DocumentSerializer` method), 263, 276

`dump()` (`app.serializers.role.RoleSerializer` method), 285, 294

`dump()` (`app.serializers.user.UserExportWordSerializer` method), 299, 314

`dump()` (`app.serializers.user.UserSerializer` method), 305, 321

`.dumps()` (`app.serializers.auth.AuthUserConfirmResetPasswordSerializer` method), 204, 216

`.dumps()` (`app.serializers.auth.AuthUserLoginSerializer` method), 210, 222

`.dumps()` (`app.serializers.core._SearchOrderSerializer` method), 246

`.dumps()` (`app.serializers.core._SearchValueSerializer` method), 252

`.dumps()` (`app.serializers.core.SearchSerializer` method), 228, 238

`.dumps()` (`app.serializers.document.DocumentAttachmentSerializer` method), 257, 269

`.dumps()` (`app.serializers.document.DocumentSerializer` method), 263, 276

`.dumps()` (`app.serializers.role.RoleSerializer` method), 285, 294

`.dumps()` (`app.serializers.user.UserExportWordSerializer` method), 300, 315

E

`either()` (`app.celery.MyCelery` method), 115, 135

`email` (`app.models.user.User` attribute), 185, 197

`email` (`database.migrations.aad_create_user_roles_table._OldUser` attribute), 357

`enable_utc` (`config.Config` attribute), 405, 430

`enable_utc` (`config.DevConfig` attribute), 412, 432

`enable_utc` (`config.ProdConfig` attribute), 420, 434

`enable_utc` (`config.TestConfig` attribute), 428, 435

`ensure_password()` (`app.models.user.User` static method), 190, 197

`errno` (`app.exceptions.FileEmptyError` attribute), 140

`error_messages` (`app.serializers.auth.AuthUserConfirmResetPasswordSerializer` attribute), 202, 217

`error_messages` (`app.serializers.auth.AuthUserLoginSerializer` attribute), 208, 223

`error_messages` (`app.serializers.core._SearchOrderSerializer` attribute), 246

`error_messages` (`app.serializers.core._SearchValueSerializer` attribute), 252

`error_messages` (`app.serializers.core.SearchSerializer` attribute), 226, 238

`error_messages` (`app.serializers.document.DocumentAttachmentSerializer` attribute), 256, 270

`error_messages` (`app.serializers.document.DocumentSerializer` attribute), 261, 276

`error_messages` (`app.serializers.role.RoleSerializer` attribute), 283, 294

`error_messages` (`app.serializers.user.UserExportWordSerializer` attribute), 298, 315

`error_messages` (`app.serializers.user.UserSerializer` attribute), 304, 321

`events` (`app.celery.MyCelery` attribute), 107, 135

`events` (`app.celery.MyCelery` attribute), 107, 135

`expires` (`app.celery.ContextTask` attribute), 90, 124

`export_user_data_in_excel()` (`app.services.task.TaskService` method), 334

`export_user_data_in_excel_and_word()` (`app.services.task.TaskService` method), 334

`export_user_data_in_excel_task()` (in module `app.celery.excel.tasks`), 85

`export_user_data_in_word()` (`app.services.task.TaskService` method), 334

`export_user_data_in_word_task()` (in module `app.celery.word.tasks`), 87

`ExportUsersExcelAndWordResource` (class in `app.blueprints.users`), 61, 79

- ExportUsersExcelResource (class in *app.blueprints.users*), 64, 80
- ExportUsersWordResource (class in *app.blueprints.users*), 67, 80
- ## F
- Factory (class in *database.factories*), 348, 350
- factory() (in module *tests.conftest*), 390, 391
- fail() (*app.serializers.core.TimestampField* method), 233, 242
- fail() (*app.serializers.role.RoleName* method), 282, 290
- fail() (*app.serializers.user.VerifyRoleId* method), 311, 325
- fields (*app.serializers.auth.AuthUserConfirmResetPasswordSerializer* attribute), 217
- fields (*app.serializers.auth.AuthUserLoginSerializer* attribute), 223
- fields (*app.serializers.core._SearchOrderSerializer* attribute), 246
- fields (*app.serializers.core._SearchValueSerializer* attribute), 252
- fields (*app.serializers.core.SearchSerializer* attribute), 238
- fields (*app.serializers.document.DocumentAttachmentSerializer* attribute), 270
- fields (*app.serializers.document.DocumentSerializer* attribute), 276
- fields (*app.serializers.role.RoleSerializer* attribute), 294
- fields (*app.serializers.user.UserExportWordSerializer* attribute), 315
- fields (*app.serializers.user.UserSerializer* attribute), 321
- FileEmptyError, 140
- filename (*app.exceptions.FileEmptyError* attribute), 140
- filename2 (*app.exceptions.FileEmptyError* attribute), 140
- FileStorage (class in *app.utils.file_storage*), 339, 340
- filter() (*app.models.base.Base* class method), 155, 159
- filter() (*app.models.document.Document* class method), 166, 169
- filter() (*app.models.role.Role* class method), 177, 181
- filter() (*app.models.user.User* class method), 190, 197
- filter() (*database.migrations.aad_create_user_roles_table._OldUser* class method), 357
- filter() (*database.migrations.aaf_remove_role_slug_column._OldRole* class method), 361
- filter() (*database.migrations.Migration* class method), 367, 372
- filter_by_keys() (in module *app.utils*), 345, 346
- finalize() (*app.celery.MyCelery* method), 116, 135
- find() (*app.managers.base.BaseManager* method), 142, 143
- find() (*app.managers.document.DocumentManager* method), 144
- find() (*app.managers.role.RoleManager* method), 146
- find() (*app.managers.user.UserManager* method), 148
- find() (*app.services.base.BaseService* method), 329
- find() (*app.services.document.DocumentService* method), 331
- find() (*app.services.role.RoleService* method), 332, 333
- find() (*app.services.user.UserService* method), 336
- find_by_email() (*app.managers.user.UserManager* method), 148
- find_by_id() (*app.services.task.TaskService* method), 334
- find_longest_word() (in module *app.utils*), 345, 346
- FLASK_RESTFUL_PREFIX (*config.Config* attribute), 402, 429
- FLASK_RESTFUL_PREFIX (*config.DevConfig* attribute), 409, 431
- FLASK_RESTFUL_PREFIX (*config.ProdConfig* attribute), 417, 433
- FLASK_RESTFUL_PREFIX (*config.TestConfig* attribute), 425, 434
- from_config (*app.celery.ContextTask* attribute), 90, 124
- from_dict() (*app.serializers.auth.AuthUserConfirmResetPasswordSerializer* class method), 204, 217
- from_dict() (*app.serializers.auth.AuthUserLoginSerializer* class method), 210, 223
- from_dict() (*app.serializers.core._SearchOrderSerializer* class method), 246
- from_dict() (*app.serializers.core._SearchValueSerializer* class method), 252
- from_dict() (*app.serializers.core.SearchSerializer* class method), 228, 238
- from_dict() (*app.serializers.document.DocumentAttachmentSerializer* class method), 257, 270
- from_dict() (*app.serializers.document.DocumentSerializer* class method), 263, 276
- from_dict() (*app.serializers.role.RoleSerializer* class method), 285, 294
- from_dict() (*app.serializers.user.UserExportWordSerializer* class method), 300, 315
- from_dict() (*app.serializers.user.UserSerializer* class method), 305, 321
- fs_uniquifier (*app.models.user.User* attribute), 185, 197
- ## G
- gen_task_name() (*app.celery.MyCelery* method), 116, 135
- genre (*app.models.user.User* attribute), 185, 197
- genre (*database.migrations.aad_create_user_roles_table._OldUser* attribute), 358

- [get\(\)](#) ([app.blueprints.auth.ResetPasswordResource method](#)), 20, 23
[get\(\)](#) ([app.blueprints.base.WelcomeResource method](#)), 27, 29
[get\(\)](#) ([app.blueprints.documents.DocumentResource method](#)), 34, 40
[get\(\)](#) ([app.blueprints.roles.RoleResource method](#)), 49, 54
[get\(\)](#) ([app.blueprints.tasks.TaskStatusResource method](#)), 59, 60
[get\(\)](#) ([app.blueprints.users.UserResource method](#)), 77, 83
[get\(\)](#) ([app.managers.base.BaseManager method](#)), 142, 143
[get\(\)](#) ([app.managers.document.DocumentManager method](#)), 144
[get\(\)](#) ([app.managers.role.RoleManager method](#)), 146
[get\(\)](#) ([app.managers.user.UserManager method](#)), 148
[get\(\)](#) ([app.models.base.Base class method](#)), 155, 159
[get\(\)](#) ([app.models.document.Document class method](#)), 166, 170
[get\(\)](#) ([app.models.role.Role class method](#)), 177, 181
[get\(\)](#) ([app.models.user.User class method](#)), 191, 197
[get\(\)](#) ([app.services.base.BaseService method](#)), 329
[get\(\)](#) ([app.services.document.DocumentService method](#)), 331
[get\(\)](#) ([app.services.role.RoleService method](#)), 332, 333
[get\(\)](#) ([app.services.user.UserService method](#)), 336
[get\(\)](#) ([database.migrations.aad_create_user_roles_table._OldUser class method](#)), 358
[get\(\)](#) ([database.migrations.aaf_remove_role_slug_column._OldRole class method](#)), 361
[get\(\)](#) ([database.migrations.Migration class method](#)), 368, 372
[get\(\)](#) ([tests.custom_flask_client.CustomFlaskClient method](#)), 393, 395
[get_attr_from_module\(\)](#) (in module [app.utils](#)), 345, 346
[get_attribute\(\)](#) ([app.serializers.auth.AuthUserConfirmResetPasswordSerializer method](#)), 204, 217
[get_attribute\(\)](#) ([app.serializers.auth.AuthUserLoginSerializer method](#)), 210, 223
[get_attribute\(\)](#) ([app.serializers.core._SearchOrderSerializer method](#)), 246
[get_attribute\(\)](#) ([app.serializers.core._SearchValueSerializer method](#)), 252
[get_attribute\(\)](#) ([app.serializers.core.SearchSerializer method](#)), 228, 238
[get_attribute\(\)](#) ([app.serializers.document.DocumentAttachmentSerializer method](#)), 258, 270
[get_attribute\(\)](#) ([app.serializers.document.DocumentSerializer method](#)), 264, 277
[get_attribute\(\)](#) ([app.serializers.role.RoleSerializer method](#)), 286, 295
[get_attribute\(\)](#) ([app.serializers.user.UserExportWordSerializer method](#)), 300, 315
[get_attribute\(\)](#) ([app.serializers.user.UserSerializer method](#)), 306, 322
[get_auth_token\(\)](#) ([app.models.user.User method](#)), 191, 197
[get_basename\(\)](#) ([app.utils.file_storage.FileStorage static method](#)), 339, 340
[get_blueprints\(\)](#) (in module [app.blueprints](#)), 84
[get_by_id\(\)](#) ([app.models.base.Base class method](#)), 155, 159
[get_by_id\(\)](#) ([app.models.document.Document class method](#)), 166, 170
[get_by_id\(\)](#) ([app.models.role.Role class method](#)), 177, 181
[get_by_id\(\)](#) ([app.models.user.User class method](#)), 191, 197
[get_by_id\(\)](#) ([database.migrations.aad_create_user_roles_table._OldUser class method](#)), 358
[get_by_id\(\)](#) ([database.migrations.aaf_remove_role_slug_column._OldRole class method](#)), 361
[get_by_id\(\)](#) ([database.migrations.Migration class method](#)), 368, 372
[get_db_models\(\)](#) (in module [app.models](#)), 200
[get_document_content\(\)](#) ([app.services.document.DocumentService method](#)), 331
[get_fields\(\)](#) ([app.models.base.Base class method](#)), 156, 159
[get_fields\(\)](#) ([app.models.document.Document class method](#)), 166, 170
[get_fields\(\)](#) ([app.models.role.Role class method](#)), 177, 181
[get_fields\(\)](#) ([app.models.user.User class method](#)), 191, 197
[get_fields\(\)](#) ([database.migrations.aad_create_user_roles_table._OldUser class method](#)), 358
[get_fields\(\)](#) ([database.migrations.aaf_remove_role_slug_column._OldRole class method](#)), 361
[get_filepath\(\)](#) ([app.models.document.Document method](#)), 166, 170
[get_filesize\(\)](#) ([app.utils.file_storage.FileStorage static method](#)), 339, 340
[get_id\(\)](#) ([app.models.base.Base method](#)), 156, 159
[get_id\(\)](#) ([app.models.document.Document method](#)), 166, 170
[get_id\(\)](#) ([app.models.role.Role method](#)), 177, 181
[get_id\(\)](#) ([app.models.user.User method](#)), 191, 197
[get_id\(\)](#) ([database.migrations.aad_create_user_roles_table._OldUser method](#)), 358
[get_id\(\)](#) ([database.migrations.aaf_remove_role_slug_column._OldRole method](#)), 361
[get_id\(\)](#) ([database.migrations.Migration method](#)), 368, 372

get_last_record() (*app.managers.user.UserManager* method), 148
 get_migration_names() (in module *database.migrations*), 370, 372
 get_or_create() (*app.models.base.Base* class method), 156, 159
 get_or_create() (*app.models.document.Document* class method), 166, 170
 get_or_create() (*app.models.role.Role* class method), 177, 181
 get_or_create() (*app.models.user.User* class method), 191, 197
 get_or_create() (*database.migrations.aad_create_user_roles_table.OldUser* class method), 358
 get_or_create() (*database.migrations.aaf_remove_role_slug_column.OldRole* class method), 361
 get_or_create() (*database.migrations.Migration* class method), 368, 372
 get_or_none() (*app.models.base.Base* class method), 156, 159
 get_or_none() (*app.models.document.Document* class method), 166, 170
 get_or_none() (*app.models.role.Role* class method), 177, 181
 get_or_none() (*app.models.user.User* class method), 191, 197
 get_or_none() (*database.migrations.aad_create_user_roles_table.OldUser* class method), 358
 get_or_none() (*database.migrations.aaf_remove_role_slug_column.OldRole* class method), 361
 get_or_none() (*database.migrations.Migration* class method), 368, 372
 get_permissions() (*app.models.role.Role* method), 177, 181
 get_permissions() (*database.migrations.aaf_remove_role_slug_column.OldRole* method), 361
 get_redirect_qparams() (*app.models.user.User* method), 192, 197
 get_request_file() (in module *app.utils*), 345, 346
 get_request_query_fields() (*app.utils.request_query_operator.RequestQueryOperator* static method), 343, 344
 get_reset_token() (*app.models.user.User* method), 192, 198
 get_security_payload() (*app.models.user.User* method), 192, 198
 get_seeders() (in module *database.seeds*), 376
 get_value() (*app.serializers.core.TimestampField* method), 233, 242
 get_value() (*app.serializers.role.RoleName* method), 282, 290
 get_value() (*app.serializers.user.VerifyRoleId* method), 311, 325
 GroupResult (*app.celery.MyCelery* attribute), 104, 129

H

handle_error() (*app.serializers.auth.AuthUserConfirmResetPasswordSerializer* method), 205, 217
 handle_error() (*app.serializers.auth.AuthUserLoginSerializer* method), 211, 223
 handle_error() (*app.serializers.core._SearchOrderSerializer* method), 246
 handle_error() (*app.serializers.core._SearchValueSerializer* method), 252
 handle_error() (*app.serializers.core.SearchSerializer* method), 229, 238
 handle_error() (*app.serializers.document.DocumentAttachmentSerializer* method), 258, 270
 handle_error() (*app.serializers.document.DocumentSerializer* method), 264, 277
 handle_error() (*app.serializers.role.RoleSerializer* method), 286, 295
 handle_error() (*app.serializers.user.UserExportWordSerializer* method), 301, 315
 handle_error() (*app.serializers.user.UserSerializer* method), 306, 322
 has_permission() (*app.models.user.User* method), 192, 198
 has_role() (*app.models.user.User* method), 192, 198
 head() (*tests.custom_flask_client.CustomFlaskClient* method), 393, 395
 Helper (class in *app.utils.request_query_operator*), 341, 343
 HOME (*config.Config* attribute), 402, 430
 HOME (*config.DevConfig* attribute), 410, 431
 HOME (*config.ProdConfig* attribute), 418, 433
 HOME (*config.TestConfig* attribute), 425, 434

I

id (*app.models.base.Base* attribute), 151, 159
 id (*app.models.document.Document* attribute), 161, 170
 id (*app.models.role.Role* attribute), 172, 181
 id (*app.models.user.User* attribute), 185, 198
 id (*database.migrations.aad_create_user_roles_table.OldUser* attribute), 358
 id (*database.migrations.aaf_remove_role_slug_column.OldRole* attribute), 361
 id (*database.migrations.Migration* attribute), 364, 372
 ignore_keys() (in module *app.utils*), 346
 ignore_result (*app.celery.ContextTask* attribute), 90, 124
 include (*config.Config* attribute), 405, 430
 include (*config.DevConfig* attribute), 413, 432
 include (*config.ProdConfig* attribute), 421, 434
 include (*config.TestConfig* attribute), 428, 435
 index() (*app.models.base.Base* class method), 156, 159
 index() (*app.models.document.Document* class method), 167, 170
 index() (*app.models.role.Role* class method), 178, 181

[index\(\)](#) (*app.models.user.User* class method), 192, 198
[index\(\)](#) (*database.migrations.aad_create_user_roles_table._OldUser* class method), 358
[index\(\)](#) (*database.migrations.aaf_remove_role_slug_column._OldRole* class method), 361
[index\(\)](#) (*database.migrations.Migration* class method), 368, 372
[init_app\(\)](#) (in module *app.exceptions*), 139, 140
[init_app\(\)](#) (in module *app.extensions*), 141
[init_database\(\)](#) (in module *database*), 376, 377
[init_migrations\(\)](#) (in module *database.migrations*), 371, 372
[init_seed\(\)](#) (in module *database.seeds*), 376
[insert\(\)](#) (*app.models.base.Base* class method), 156, 159
[insert\(\)](#) (*app.models.document.Document* class method), 167, 170
[insert\(\)](#) (*app.models.role.Role* class method), 178, 181
[insert\(\)](#) (*app.models.user.User* class method), 193, 198
[insert\(\)](#) (*database.migrations.aad_create_user_roles_table._OldUser* class method), 358
[insert\(\)](#) (*database.migrations.aaf_remove_role_slug_column._OldRole* class method), 361
[insert\(\)](#) (*database.migrations.Migration* class method), 368, 372
[insert_from\(\)](#) (*app.models.base.Base* class method), 156, 159
[insert_from\(\)](#) (*app.models.document.Document* class method), 167, 170
[insert_from\(\)](#) (*app.models.role.Role* class method), 178, 181
[insert_from\(\)](#) (*app.models.user.User* class method), 193, 198
[insert_from\(\)](#) (*database.migrations.aad_create_user_roles_table._OldUser* class method), 358
[insert_from\(\)](#) (*database.migrations.aaf_remove_role_slug_column._OldRole* class method), 361
[insert_from\(\)](#) (*database.migrations.Migration* class method), 368, 372
[insert_many\(\)](#) (*app.models.base.Base* class method), 156, 159
[insert_many\(\)](#) (*app.models.document.Document* class method), 167, 170
[insert_many\(\)](#) (*app.models.role.Role* class method), 178, 181
[insert_many\(\)](#) (*app.models.user.User* class method), 193, 198
[insert_many\(\)](#) (*database.migrations.aad_create_user_roles_table._OldUser* class method), 358
[insert_many\(\)](#) (*database.migrations.aaf_remove_role_slug_column._OldRole* class method), 361
[insert_many\(\)](#) (*database.migrations.Migration* class method), 368, 372
[insert_task_record\(\)](#) (in module *app.celery.tests.tasks*), 86
[is_authenticated\(\)](#) (*app.models.document.Document* attribute), 161, 170
[is_authenticated\(\)](#) (*app.models.user.User* property), 185, 198
[is_active\(\)](#) (*database.migrations.aad_create_user_roles_table._OldUser* property), 358
[is_alias\(\)](#) (*app.models.base.Base* method), 156, 159
[is_alias\(\)](#) (*app.models.document.Document* method), 167, 170
[is_alias\(\)](#) (*app.models.role.Role* method), 178, 181
[is_alias\(\)](#) (*app.models.user.User* method), 193, 198
[is_alias\(\)](#) (*database.migrations.aad_create_user_roles_table._OldUser* method), 358
[is_alias\(\)](#) (*database.migrations.aaf_remove_role_slug_column._OldRole* method), 361
[is_alias\(\)](#) (*database.migrations.Migration* method), 369, 372
[is_anonymous\(\)](#) (*app.models.user.User* property), 185, 198
[is_anonymous\(\)](#) (*database.migrations.aad_create_user_roles_table._OldUser* property), 358
[is_authenticated\(\)](#) (*app.models.user.User* property), 185, 198
[is_authenticated\(\)](#) (*database.migrations.aad_create_user_roles_table._OldUser* property), 358
[is_dirty\(\)](#) (*app.models.base.Base* method), 157, 159
[is_dirty\(\)](#) (*app.models.document.Document* method), 167, 170
[is_dirty\(\)](#) (*app.models.role.Role* method), 178, 181
[is_dirty\(\)](#) (*app.models.user.User* method), 193, 198
[is_dirty\(\)](#) (*database.migrations.aad_create_user_roles_table._OldUser* method), 358
[is_dirty\(\)](#) (*database.migrations.aaf_remove_role_slug_column._OldRole* method), 361
[is_dirty\(\)](#) (*database.migrations.Migration* method), 369, 372
[IS_macOS](#) (*app.celery.MyCelery* attribute), 105, 130
[IS_WINDOWS](#) (*app.celery.MyCelery* attribute), 104, 130

J

[jsonify\(\)](#) (*app.serializers.auth.AuthUserConfirmResetPasswordSerializer* method), 205, 217
[jsonify\(\)](#) (*app.serializers.auth.AuthUserLoginSerializer* method), 211, 223
[jsonify\(\)](#) (*app.serializers.core._SearchOrderSerializer* method), 247
[jsonify\(\)](#) (*app.serializers.core._SearchValueSerializer* method), 253
[jsonify\(\)](#) (*app.serializers.core.SearchSerializer* method), 229, 239
[jsonify\(\)](#) (*app.serializers.document.DocumentAttachmentSerializer* method), 258, 270
[jsonify\(\)](#) (*app.serializers.document.DocumentSerializer* method), 264, 277

- [jsonify\(\)](#) (*app.serializers.role.RoleSerializer* method), 286, 295
[jsonify\(\)](#) (*app.serializers.user.UserExportWordSerializer* method), 301, 316
[jsonify\(\)](#) (*app.serializers.user.UserSerializer* method), 306, 322
- ## L
- [label](#) (*app.models.role.Role* attribute), 172, 181
[last_name](#) (*app.models.user.User* attribute), 185, 198
[last_name](#) (*database.migrations.aad_create_user_roles_table* attribute), 358
[libreoffice_exec\(\)](#) (in module *app.utils.libreoffice*), 340, 341
[LibreOfficeError](#), 341
[load\(\)](#) (*app.serializers.auth.AuthUserConfirmResetPasswordSerializer* method), 205, 218
[load\(\)](#) (*app.serializers.auth.AuthUserLoginSerializer* method), 211, 224
[load\(\)](#) (*app.serializers.core._SearchOrderSerializer* method), 247
[load\(\)](#) (*app.serializers.core._SearchValueSerializer* method), 253
[load\(\)](#) (*app.serializers.core.SearchSerializer* method), 229, 239
[load\(\)](#) (*app.serializers.document.DocumentAttachmentSerializer* method), 259, 271
[load\(\)](#) (*app.serializers.document.DocumentSerializer* method), 264, 277
[load\(\)](#) (*app.serializers.role.RoleSerializer* method), 286, 295
[load\(\)](#) (*app.serializers.user.UserExportWordSerializer* method), 301, 316
[load\(\)](#) (*app.serializers.user.UserSerializer* method), 307, 322
[loader](#) (*app.celery.MyCelery* attribute), 107, 136
[loader_cls](#) (*app.celery.MyCelery* attribute), 107, 136
[loads\(\)](#) (*app.serializers.auth.AuthUserConfirmResetPasswordSerializer* method), 206, 218
[loads\(\)](#) (*app.serializers.auth.AuthUserLoginSerializer* method), 212, 224
[loads\(\)](#) (*app.serializers.core._SearchOrderSerializer* method), 247
[loads\(\)](#) (*app.serializers.core._SearchValueSerializer* method), 253
[loads\(\)](#) (*app.serializers.core.SearchSerializer* method), 230, 239
[loads\(\)](#) (*app.serializers.document.DocumentAttachmentSerializer* method), 259, 271
[loads\(\)](#) (*app.serializers.document.DocumentSerializer* method), 265, 278
[loads\(\)](#) (*app.serializers.role.RoleSerializer* method), 287, 296
[loads\(\)](#) (*app.serializers.user.UserExportWordSerializer* method), 302, 316
[loads\(\)](#) (*app.serializers.user.UserSerializer* method), 307, 323
[log](#) (*app.celery.MyCelery* attribute), 107, 136
[log_cls](#) (*app.celery.MyCelery* attribute), 107, 136
[LOG_DIRECTORY](#) (*config.Config* attribute), 402, 430
[LOG_DIRECTORY](#) (*config.DevConfig* attribute), 410, 431
[LOG_DIRECTORY](#) (*config.ProdConfig* attribute), 418, 433
[LOG_DIRECTORY](#) (*config.TestConfig* attribute), 425, 434
[LOGIN_DISABLED](#) (*config.Config* attribute), 402, 430
[LOGIN_DISABLED](#) (*config.DevConfig* attribute), 410, 431
[LOGIN_DISABLED](#) (*config.ProdConfig* attribute), 418, 433
[LOGIN_DISABLED](#) (*config.TestConfig* attribute), 425, 434
[login_user\(\)](#) (*app.services.auth.AuthService* method), 328
[logout_user\(\)](#) (*app.services.auth.AuthService* static method), 328
- ## M
- [MAIL_PASSWORD](#) (*config.Config* attribute), 402, 430
[MAIL_PASSWORD](#) (*config.DevConfig* attribute), 410, 431
[MAIL_PASSWORD](#) (*config.ProdConfig* attribute), 418, 433
[MAIL_PASSWORD](#) (*config.TestConfig* attribute), 425, 435
[MAIL_PORT](#) (*config.Config* attribute), 402, 430
[MAIL_PORT](#) (*config.DevConfig* attribute), 410, 431
[MAIL_PORT](#) (*config.ProdConfig* attribute), 418, 433
[MAIL_PORT](#) (*config.TestConfig* attribute), 425, 435
[MAIL_SERVER](#) (*config.Config* attribute), 403, 430
[MAIL_SERVER](#) (*config.DevConfig* attribute), 410, 431
[MAIL_SERVER](#) (*config.ProdConfig* attribute), 418, 433
[MAIL_SERVER](#) (*config.TestConfig* attribute), 425, 435
[MAIL_USE_SSL](#) (*config.Config* attribute), 403, 430
[MAIL_USE_SSL](#) (*config.DevConfig* attribute), 410, 431
[MAIL_USE_SSL](#) (*config.ProdConfig* attribute), 418, 433
[MAIL_USE_SSL](#) (*config.TestConfig* attribute), 426, 435
[MAIL_USE_TLS](#) (*config.Config* attribute), 403, 430
[MAIL_USE_TLS](#) (*config.DevConfig* attribute), 410, 431
[MAIL_USE_TLS](#) (*config.ProdConfig* attribute), 418, 433
[MAIL_USE_TLS](#) (*config.TestConfig* attribute), 426, 435
[MAIL_USERNAME](#) (*config.Config* attribute), 403, 430
[MAIL_USERNAME](#) (*config.DevConfig* attribute), 410, 431
[MAIL_USERNAME](#) (*config.ProdConfig* attribute), 418, 433
[MAIL_USERNAME](#) (*config.TestConfig* attribute), 425, 435
[main](#) (*app.celery.MyCelery* attribute), 107, 136
[make\(\)](#) (*database.factories.Factory* method), 348, 350, 352
[make_celery\(\)](#) (in module *app.celery*), 120, 139
[make_error\(\)](#) (*app.serializers.core.TimestampField* method), 234, 242
[make_error\(\)](#) (*app.serializers.role.RoleName* method), 282, 290
[make_error\(\)](#) (*app.serializers.user.VerifyRoleId* method), 311, 326

`make_object()` (`app.serializers.auth.AuthUserConfirmResetPasswordSerializer` method), 206, 218
`make_object()` (`app.serializers.auth.AuthUserLoginSerializer` method), 212, 224
`make_request()` (`tests.custom_flask_client.CustomFlaskClient` method), 393, 395
`map()` (`app.celery.ContextTask` method), 97, 124
`max_retries` (`app.celery.ContextTask` attribute), 90, 124
`Meta` (class in `config`), 414, 432
`method_decorators` (`app.blueprints.auth.AuthBaseResource` attribute), 10, 20
`method_decorators` (`app.blueprints.auth.AuthUserLoginResource` attribute), 12, 21
`method_decorators` (`app.blueprints.auth.AuthUserLogoutResource` attribute), 14, 22
`method_decorators` (`app.blueprints.auth.RequestResetPasswordResource` attribute), 16, 22
`method_decorators` (`app.blueprints.auth.ResetPasswordResource` attribute), 19, 23
`method_decorators` (`app.blueprints.base.BaseResource` attribute), 24, 28
`method_decorators` (`app.blueprints.base.WelcomeResource` attribute), 26, 29
`method_decorators` (`app.blueprints.documents.DocumentBaseResource` attribute), 30, 40
`method_decorators` (`app.blueprints.documents.DocumentResource` attribute), 33, 40
`method_decorators` (`app.blueprints.documents.NewDocumentResource` attribute), 36, 41
`method_decorators` (`app.blueprints.documents.SearchDocumentResource` attribute), 38, 42
`method_decorators` (`app.blueprints.roles.NewRoleResource` attribute), 43, 52
`method_decorators` (`app.blueprints.roles.RoleBaseResource` attribute), 46, 53
`method_decorators` (`app.blueprints.roles.RoleResource` attribute), 48, 54
`method_decorators` (`app.blueprints.roles.RolesSearchResource` attribute), 51, 54
`method_decorators` (`app.blueprints.tasks.TaskResource` attribute), 56, 59
`method_decorators` (`app.blueprints.tasks.TaskStatusResource` attribute), 58, 60
`method_decorators` (`app.blueprints.users.ExportUsersExcelAndWordResource` attribute), 62, 79
`method_decorators` (`app.blueprints.users.ExportUsersExcelResource` attribute), 65, 80
`method_decorators` (`app.blueprints.users.ExportUsersWordResource` attribute), 68, 81
`method_decorators` (`app.blueprints.users.NewUserResource` attribute), 70, 82
`method_decorators` (`app.blueprints.users.UserBaseResource` attribute), 73, 82
`method_decorators` (`app.blueprints.users.UserResource` attribute), 75, 83
`method_decorators` (`app.blueprints.users.UsersSearchResource` attribute), 78, 84
`methods` (`app.blueprints.auth.AuthBaseResource` attribute), 10, 21
`methods` (`app.blueprints.auth.AuthUserLoginResource` attribute), 12, 21
`methods` (`app.blueprints.auth.AuthUserLogoutResource` attribute), 14, 22
`methods` (`app.blueprints.auth.RequestResetPasswordResource` attribute), 16, 22
`methods` (`app.blueprints.auth.ResetPasswordResource` attribute), 19, 23
`methods` (`app.blueprints.base.BaseResource` attribute), 24, 28
`methods` (`app.blueprints.base.WelcomeResource` attribute), 26, 29
`methods` (`app.blueprints.documents.DocumentBaseResource` attribute), 30, 40
`methods` (`app.blueprints.documents.DocumentResource` attribute), 33, 40
`methods` (`app.blueprints.documents.NewDocumentResource` attribute), 36, 41
`methods` (`app.blueprints.documents.SearchDocumentResource` attribute), 38, 42
`methods` (`app.blueprints.roles.NewRoleResource` attribute), 43, 52
`methods` (`app.blueprints.roles.RoleBaseResource` attribute), 46, 53
`methods` (`app.blueprints.roles.RoleResource` attribute), 48, 54
`methods` (`app.blueprints.roles.RolesSearchResource` attribute), 51, 54
`methods` (`app.blueprints.tasks.TaskResource` attribute), 56, 59
`methods` (`app.blueprints.tasks.TaskStatusResource` attribute), 58, 60
`methods` (`app.blueprints.users.ExportUsersExcelAndWordResource` attribute), 62, 79
`methods` (`app.blueprints.users.ExportUsersExcelResource` attribute), 65, 80
`methods` (`app.blueprints.users.ExportUsersWordResource` attribute), 68, 81
`methods` (`app.blueprints.users.NewUserResource` attribute), 70, 82
`methods` (`app.blueprints.users.UserBaseResource` attribute), 73, 82
`methods` (`app.blueprints.users.UserResource` attribute), 75, 83
`methods` (`app.blueprints.users.UsersSearchResource` attribute), 78, 84
`Middleware` (class in `app.middleware`), 149, 150
`migrate_actions()` (in module `database.migrations`), 371, 372

Migration (class in `database.migrations`), 363, 371
mime_type (app.models.document.Document attribute), 161, 170
missing (app.serializers.core.TimestampField property), 232, 242
missing (app.serializers.role.RoleName property), 280, 290
missing (app.serializers.user.VerifyRoleId property), 309, 326
MOCKUP_DIRECTORY (config.Config attribute), 403, 430
MOCKUP_DIRECTORY (config.DevConfig attribute), 410, 431
MOCKUP_DIRECTORY (config.ProdConfig attribute), 418, 433
MOCKUP_DIRECTORY (config.TestConfig attribute), 426, 435
module
 app, 8
 app.blueprints, 8
 app.blueprints.auth, 9
 app.blueprints.base, 23
 app.blueprints.documents, 29
 app.blueprints.roles, 42
 app.blueprints.tasks, 55
 app.blueprints.users, 61
 app.celery, 84
 app.celery.excel, 85
 app.celery.excel.tasks, 85
 app.celery.tasks, 86
 app.celery.tests, 86
 app.celery.tests.tasks, 86
 app.celery.word, 87
 app.celery.word.tasks, 87
 app.exceptions, 139
 app.extensions, 140
 app.managers, 141
 app.managers.base, 141
 app.managers.document, 143
 app.managers.role, 145
 app.managers.user, 146
 app.middleware, 149
 app.models, 150
 app.models.base, 151
 app.models.document, 160
 app.models.role, 171
 app.models.user, 182
 app.models.user_roles, 200
 app.serializers, 200
 app.serializers.auth, 201
 app.serializers.core, 225
 app.serializers.document, 254
 app.serializers.role, 279
 app.serializers.user, 297
 app.services, 326

app.services.auth, 327
app.services.base, 328
app.services.document, 330
app.services.role, 331
app.services.task, 333
app.services.user, 335
app.swagger, 336
app.swagger.auth, 337
app.swagger.core, 337
app.swagger.document, 337
app.swagger.role, 337
app.swagger.user, 337
app.utils, 337
app.utils.constants, 337
app.utils.decorators, 338
app.utils.file_storage, 338
app.utils.libreoffice, 340
app.utils.request_query_operator, 341
config, 399
database, 348
database.factories, 348
database.migrations, 352
database.migrations.aaa_add_genre_column_on_user_table, 353
database.migrations.aab_add_created_by_column_on_user, 354
database.migrations.aac_create_documents_table, 355
database.migrations.aad_create_user_roles_table, 356
database.migrations.aaf_remove_role_slug_column, 359
database.migrations.aag_add_fs_uniquifier_column_on_us, 362
database.seeds, 373
database.seeds.document_seeder, 373
database.seeds.role_seeder, 374
database.seeds.user_seeder, 375
tests, 377
tests.blueprints, 379
tests.blueprints.test_auth, 379
tests.blueprints.test_base, 380
tests.blueprints.test_documents, 381
tests.blueprints.test_roles, 383
tests.blueprints.test_tasks, 384
tests.blueprints.test_users, 385
tests.celery, 387
tests.celery.test_celery, 387
tests.celery.test_excel, 388
tests.celery.test_tasks, 388
tests.celery.test_word, 389
tests.conftest, 390
tests.custom_flask_client, 391
tests.test_config, 397

tests.test_db, 397
 tests.test_mail, 397
 tests.test_middleware, 398
 mro() (*config.Meta* method), 415, 433
 MS_EXCEL_MIME_TYPE (in module *app.utils.constants*), 338
 MyCelery (class in *app.celery*), 102, 129

N

name (*app.celery.ContextTask* attribute), 90, 124
 name (*app.models.document.Document* attribute), 162, 170
 name (*app.models.role.Role* attribute), 172, 181
 name (*app.models.user.User* attribute), 185, 198
 name (*app.serializers.core.TimestampField* attribute), 232, 242
 name (*app.serializers.role.RoleName* attribute), 280, 290
 name (*app.serializers.user.VerifyRoleId* attribute), 309, 326
 name (*database.migrations.aad_create_user_roles_table._OldUser* attribute), 358
 name (*database.migrations.aaf_remove_role_slug_column._OldRole* attribute), 361
 name (*database.migrations.Migration* attribute), 364, 372
 name (*database.seeds.document_seeder.DocumentSeeder* attribute), 373, 374
 name (*database.seeds.role_seeder.RoleSeeder* attribute), 374, 375
 name (*database.seeds.user_seeder.UserSeeder* attribute), 375, 376
 NewDocumentResource (class in *app.blueprints.documents*), 34, 40
 NewRoleResource (class in *app.blueprints.roles*), 42, 52
 NewUserResource (class in *app.blueprints.users*), 69, 81
 noop() (*app.models.base.Base* class method), 157, 159
 noop() (*app.models.document.Document* class method), 167, 170
 noop() (*app.models.role.Role* class method), 178, 181
 noop() (*app.models.user.User* class method), 193, 198
 noop() (*database.migrations.aad_create_user_roles_table._OldUser* class method), 358
 noop() (*database.migrations.aaf_remove_role_slug_column._OldRole* class method), 361
 noop() (*database.migrations.Migration* class method), 369, 372
 now() (*app.celery.MyCelery* method), 116, 136

O

oid (*app.celery.MyCelery* attribute), 108, 136
 on_after_configure (*app.celery.MyCelery* attribute), 108, 136
 on_after_finalize (*app.celery.MyCelery* attribute), 108, 136
 on_after_fork (*app.celery.MyCelery* attribute), 108, 136
 on_bind_field() (*app.serializers.auth.AuthUserConfirmResetPasswordSerializer* method), 206, 218
 on_bind_field() (*app.serializers.auth.AuthUserLoginSerializer* method), 212, 224
 on_bind_field() (*app.serializers.core._SearchOrderSerializer* method), 248
 on_bind_field() (*app.serializers.core._SearchValueSerializer* method), 254
 on_bind_field() (*app.serializers.core.SearchSerializer* method), 230, 240
 on_bind_field() (*app.serializers.document.DocumentAttachmentSerializer* method), 260, 271
 on_bind_field() (*app.serializers.document.DocumentSerializer* method), 265, 278
 on_bind_field() (*app.serializers.role.RoleSerializer* method), 287, 296
 on_bind_field() (*app.serializers.user.UserExportWordSerializer* method), 302, 317
 on_bind_field() (*app.serializers.user.UserSerializer* method), 308, 323
 on_bound() (*app.celery.ContextTask* class method), 97, 124
 on_configure (*app.celery.MyCelery* attribute), 108, 136
 on_failure() (*app.celery.ContextTask* method), 98, 124
 on_init() (*app.celery.MyCelery* method), 116, 136
 on_retry() (*app.celery.ContextTask* method), 98, 125
 on_success() (*app.celery.ContextTask* method), 98, 125
 open() (*tests.custom_flask_client.CustomFlaskClient* method), 393, 395
 options() (*tests.custom_flask_client.CustomFlaskClient* method), 393, 396
 OPTIONS_CLASS (*app.serializers.auth.AuthUserConfirmResetPasswordSerializer* attribute), 214
 OPTIONS_CLASS (*app.serializers.auth.AuthUserLoginSerializer* attribute), 220
 OPTIONS_CLASS (*app.serializers.core._SearchOrderSerializer* attribute), 243
 OPTIONS_CLASS (*app.serializers.core._SearchValueSerializer* attribute), 249
 OPTIONS_CLASS (*app.serializers.core.SearchSerializer* attribute), 235
 OPTIONS_CLASS (*app.serializers.document.DocumentAttachmentSerializer* attribute), 267
 OPTIONS_CLASS (*app.serializers.document.DocumentSerializer* attribute), 272
 OPTIONS_CLASS (*app.serializers.role.RoleSerializer* attribute), 290
 OPTIONS_CLASS (*app.serializers.user.UserExportWordSerializer* attribute), 312

- OPTIONS_CLASS (*app.serializers.user.UserSerializer* attribute), 317
 opts (*app.serializers.auth.AuthUserConfirmResetPasswordSerializer* attribute), 202, 218
 opts (*app.serializers.auth.AuthUserLoginSerializer* attribute), 208, 224
 opts (*app.serializers.core._SearchOrderSerializer* attribute), 248
 opts (*app.serializers.core._SearchValueSerializer* attribute), 254
 opts (*app.serializers.core.SearchSerializer* attribute), 226, 240
 opts (*app.serializers.document.DocumentAttachmentSerializer* attribute), 256, 271
 opts (*app.serializers.document.DocumentSerializer* attribute), 261, 278
 opts (*app.serializers.role.RoleSerializer* attribute), 283, 296
 opts (*app.serializers.user.UserExportWordSerializer* attribute), 298, 317
 opts (*app.serializers.user.UserSerializer* attribute), 304, 323
 ordered (*app.serializers.document.DocumentSerializer.Meta* attribute), 272
 ordered (*app.serializers.role.RoleSerializer.Meta* attribute), 290
 ordered (*app.serializers.user.UserSerializer.Meta* attribute), 317
- ## P
- parent (*app.serializers.core.TimestampField* attribute), 232, 242
 parent (*app.serializers.role.RoleName* attribute), 280, 290
 parent (*app.serializers.user.VerifyRoleId* attribute), 310, 326
 parse_content_type() (*app.middleware.Middleware* static method), 149, 150
 parser (*app.blueprints.documents.NewDocumentResource* attribute), 36, 41
 password (*app.models.user.User* attribute), 186, 198
 password (*database.migrations.aad_create_user_roles_table* attribute), 358
 patch() (*tests.custom_flask_client.CustomFlaskClient* method), 394, 396
 Pickler (*app.celery.MyCelery* attribute), 130
 pool (*app.celery.MyCelery* property), 108, 136
 pop_request() (*app.celery.ContextTask* method), 99, 125
 pos_to_char() (in module *app.utils*), 346
 post() (*app.blueprints.auth.AuthUserLoginResource* method), 13, 21
 post() (*app.blueprints.auth.AuthUserLogoutResource* method), 15, 22
 post() (*app.blueprints.auth.RequestResetPasswordResource* method), 18, 22
 post() (*app.blueprints.auth.ResetPasswordResource* method), 20, 23
 post() (*app.blueprints.documents.NewDocumentResource* method), 37, 41
 post() (*app.blueprints.documents.SearchDocumentResource* method), 39, 42
 post() (*app.blueprints.roles.NewRoleResource* method), 45, 53
 post() (*app.blueprints.roles.RolesSearchResource* method), 52, 55
 post() (*app.blueprints.users.ExportUsersExcelAndWordResource* method), 64, 80
 post() (*app.blueprints.users.ExportUsersExcelResource* method), 66, 80
 post() (*app.blueprints.users.ExportUsersWordResource* method), 69, 81
 post() (*app.blueprints.users.NewUserResource* method), 72, 82
 post() (*app.blueprints.users.UsersSearchResource* method), 79, 84
 post() (*tests.custom_flask_client.CustomFlaskClient* method), 394, 396
 prepare_config() (*app.celery.MyCelery* method), 117, 136
 preserve_context (*tests.custom_flask_client.CustomFlaskClient* attribute), 391, 396
 priority (*app.celery.ContextTask* attribute), 90, 125
 process_input() (*app.serializers.document.DocumentAttachmentSerializer* method), 260, 271
 process_input() (*app.serializers.user.UserExportWordSerializer* method), 302, 317
 ProdConfig (class in *config*), 415, 433
 producer_or_acquire() (*app.celery.MyCelery* method), 117, 136
 producer_pool (*app.celery.MyCelery* property), 108, 136
 provide_automatic_options (*app.blueprints.auth.AuthBaseResource* attribute), 10, 21
 provide_automatic_options (*app.blueprints.auth.AuthUserLoginResource* attribute), 12, 21
 provide_automatic_options (*app.blueprints.auth.AuthUserLogoutResource* attribute), 14, 22
 provide_automatic_options (*app.blueprints.auth.RequestResetPasswordResource* attribute), 17, 22
 provide_automatic_options (*app.blueprints.auth.ResetPasswordResource* attribute), 19, 23
 provide_automatic_options

(*app.blueprints.base.BaseResource* attribute), 24, 28
provide_automatic_options (*app.blueprints.base>WelcomeResource* attribute), 26, 29
provide_automatic_options (*app.blueprints.documents.DocumentBaseResource* attribute), 30, 40
provide_automatic_options (*app.blueprints.documents.DocumentResource* attribute), 33, 40
provide_automatic_options (*app.blueprints.documents.NewDocumentResource* attribute), 36, 41
provide_automatic_options (*app.blueprints.documents.SearchDocumentResource* attribute), 38, 42
provide_automatic_options (*app.blueprints.roles.NewRoleResource* attribute), 43, 53
provide_automatic_options (*app.blueprints.roles.RoleBaseResource* attribute), 46, 53
provide_automatic_options (*app.blueprints.roles.RoleResource* attribute), 48, 54
provide_automatic_options (*app.blueprints.roles.RolesSearchResource* attribute), 51, 55
provide_automatic_options (*app.blueprints.tasks.TaskResource* attribute), 56, 60
provide_automatic_options (*app.blueprints.tasks.TaskStatusResource* attribute), 58, 60
provide_automatic_options (*app.blueprints.users.ExportUsersExcelAndWordResource* attribute), 62, 80
provide_automatic_options (*app.blueprints.users.ExportUsersExcelResource* attribute), 65, 80
provide_automatic_options (*app.blueprints.users.ExportUsersWordResource* attribute), 68, 81
provide_automatic_options (*app.blueprints.users.NewUserResource* attribute), 70, 82
provide_automatic_options (*app.blueprints.users.UserBaseResource* attribute), 73, 82
provide_automatic_options (*app.blueprints.users.UserResource* attribute), 75, 83
provide_automatic_options (*app.blueprints.users.UsersSearchResource* attribute), 78, 84
push_request() (*app.celery.ContextTask* method), 99, 125
put() (*app.blueprints.documents.DocumentResource* method), 34, 40
put() (*app.blueprints.roles.RoleResource* method), 49, 54
put() (*app.blueprints.users.UserResource* method), 77, 83
put() (*tests.custom_flask_client.CustomFlaskClient* method), 394, 396
R
rate_limit (*app.celery.ContextTask* attribute), 91, 125
raw() (*app.managers.base.BaseManager* method), 142, 143
raw() (*app.managers.document.DocumentManager* method), 144
raw() (*app.managers.role.RoleManager* method), 146
raw() (*app.managers.user.UserManager* method), 148
raw() (*app.models.base.Base* static method), 157, 159
raw() (*app.models.document.Document* static method), 167, 170
raw() (*app.models.role.Role* static method), 178, 181
raw() (*app.models.user.User* static method), 193, 198
raw() (*database.migrations.aad_create_user_roles_table._OldUser* static method), 358
raw() (*database.migrations.aaf_remove_role_slug_column._OldRole* static method), 361
raw() (*database.migrations.Migration* class method), 369, 372
register_task() (*app.celery.MyCelery* method), 117, 136
registry_cls (*app.celery.MyCelery* attribute), 109, 137
reject_on_worker_lost (*app.celery.ContextTask* attribute), 91, 125
reload() (*app.models.base.Base* method), 157, 159
reload() (*app.models.document.Document* method), 167, 170
reload() (*app.models.role.Role* method), 178, 181
reload() (*app.models.user.User* method), 193, 198
reload() (*database.migrations.aad_create_user_roles_table._OldUser* method), 358
reload() (*database.migrations.aaf_remove_role_slug_column._OldRole* method), 361
remove_permissions() (*app.models.role.Role* method), 179, 181
remove_permissions() (*database.migrations.aaf_remove_role_slug_column._OldRole* method), 361
RemoveRoleSlugColumn (class in *database.migrations.aaf_remove_role_slug_column*),

- 359
- `rename()` (`app.utils.file_storage.FileStorage` static method), 339, 340
- `replace()` (`app.celery.ContextTask` method), 99, 126
- `replace()` (`app.models.base.Base` class method), 157, 159
- `replace()` (`app.models.document.Document` class method), 168, 170
- `replace()` (`app.models.role.Role` class method), 179, 182
- `replace()` (`app.models.user.User` class method), 193, 198
- `replace()` (`database.migrations.aad_create_user_roles_table` class method), 358
- `replace()` (`database.migrations.aaf_remove_role_slug_column` class method), 361
- `replace()` (`database.migrations.Migration` class method), 369, 372
- `replace_many()` (`app.models.base.Base` class method), 157, 159
- `replace_many()` (`app.models.document.Document` class method), 168, 170
- `replace_many()` (`app.models.role.Role` class method), 179, 182
- `replace_many()` (`app.models.user.User` class method), 194, 198
- `replace_many()` (`database.migrations.aad_create_user_roles_table` class method), 358
- `replace_many()` (`database.migrations.aaf_remove_role_slug_column` class method), 361
- `replace_many()` (`database.migrations.Migration` class method), 369, 372
- `representations` (`app.blueprints.auth.AuthBaseResource` attribute), 10, 21
- `representations` (`app.blueprints.auth.AuthUserLoginResource` attribute), 12, 21
- `representations` (`app.blueprints.auth.AuthUserLogoutResource` attribute), 14, 22
- `representations` (`app.blueprints.auth.RequestResetPasswordResource` attribute), 17, 22
- `representations` (`app.blueprints.auth.ResetPasswordResource` attribute), 19, 23
- `representations` (`app.blueprints.base.BaseResource` attribute), 25, 28
- `representations` (`app.blueprints.base.WelcomeResource` attribute), 27, 29
- `representations` (`app.blueprints.documents.DocumentBaseResource` attribute), 31, 40
- `representations` (`app.blueprints.documents.DocumentResource` attribute), 33, 40
- `representations` (`app.blueprints.documents.NewDocumentResource` attribute), 36, 41
- `representations` (`app.blueprints.documents.SearchDocumentResource` attribute), 38, 42
- `representations` (`app.blueprints.roles.NewRoleResource` attribute), 43, 53
- `representations` (`app.blueprints.roles.RoleBaseResource` attribute), 46, 53
- `representations` (`app.blueprints.roles.RoleResource` attribute), 48, 54
- `representations` (`app.blueprints.roles.RolesSearchResource` attribute), 51, 55
- `representations` (`app.blueprints.tasks.TaskResource` attribute), 56, 60
- `representations` (`app.blueprints.tasks.TaskStatusResource` attribute), 58, 60
- `representations` (`app.blueprints.users.ExportUsersExcelAndWordResource` attribute), 62, 80
- `representations` (`app.blueprints.users.ExportUsersExcelResource` attribute), 65, 80
- `representations` (`app.blueprints.users.ExportUsersWordResource` attribute), 68, 81
- `representations` (`app.blueprints.users.NewUserResource` attribute), 70, 82
- `representations` (`app.blueprints.users.UserBaseResource` attribute), 73, 82
- `representations` (`app.blueprints.users.UserResource` attribute), 75, 83
- `representations` (`app.blueprints.users.UsersSearchResource` attribute), 78, 84
- `request()` (`app.celery.ContextTask` attribute), 89, 120
- `request` (`app.celery.ContextTask` property), 91, 126
- `request_headers` (`app.blueprints.auth.AuthBaseResource` attribute), 328
- `request_stack` (`app.celery.ContextTask` attribute), 91, 126
- `RequestQueryOperator` (class in `app.utils.request_query_operator`), 343, 344
- `RequestResetPasswordResource` (class in `app.blueprints.auth`), 15, 22
- `reset_password_email()` (`app.services.task.TaskService` method), 334
- `reset_password_email_task()` (in module `app.celery.tasks`), 86
- `RESET_TOKEN_EXPIRES` (`config.Config` attribute), 403, 430
- `RESET_TOKEN_EXPIRES` (`config.DevConfig` attribute), 411, 431
- `RESET_TOKEN_EXPIRES` (`config.ProdConfig` attribute), 419, 433
- `RESET_TOKEN_EXPIRES` (`config.TestConfig` attribute), 426, 435
- `ResetPasswordResource` (class in `app.blueprints.auth`), 18, 23
- `resolve_redirect()` (`tests.custom_flask_client.CustomFlaskClient` method), 328

- method*), 394, 396
 - RESTX_ERROR_404_HELP (*config.Config* attribute), 403, 430
 - RESTX_ERROR_404_HELP (*config.DevConfig* attribute), 411, 431
 - RESTX_ERROR_404_HELP (*config.ProdConfig* attribute), 419, 433
 - RESTX_ERROR_404_HELP (*config.TestConfig* attribute), 426, 435
 - RESTX_MASK_SWAGGER (*config.Config* attribute), 403, 430
 - RESTX_MASK_SWAGGER (*config.DevConfig* attribute), 411, 431
 - RESTX_MASK_SWAGGER (*config.ProdConfig* attribute), 419, 433
 - RESTX_MASK_SWAGGER (*config.TestConfig* attribute), 426, 435
 - result_backend (*config.Config* attribute), 405, 430
 - result_backend (*config.DevConfig* attribute), 413, 432
 - result_backend (*config.ProdConfig* attribute), 421, 434
 - result_backend (*config.TestConfig* attribute), 428, 435
 - result_expires (*config.Config* attribute), 405, 430
 - result_expires (*config.DevConfig* attribute), 413, 432
 - result_expires (*config.ProdConfig* attribute), 421, 434
 - result_expires (*config.TestConfig* attribute), 428, 435
 - result_extended (*config.Config* attribute), 405, 431
 - result_extended (*config.DevConfig* attribute), 413, 432
 - result_extended (*config.ProdConfig* attribute), 421, 434
 - result_extended (*config.TestConfig* attribute), 428, 435
 - result_serializer (*config.Config* attribute), 406, 431
 - result_serializer (*config.DevConfig* attribute), 413, 432
 - result_serializer (*config.ProdConfig* attribute), 421, 434
 - result_serializer (*config.TestConfig* attribute), 428, 435
 - resultrepr_maxsize (*app.celery.ContextTask* attribute), 91, 126
 - ResultSet (*app.celery.MyCelery* attribute), 105, 130
 - retry() (*app.celery.ContextTask* method), 99, 126
 - Role (*class in app.models.role*), 171, 180
 - role (*database.migrations.aad_create_user_roles_table.OldUser* attribute), 358
 - role_id (*database.migrations.aad_create_user_roles_table.OldUser* attribute), 358
 - role_serializer (*app.blueprints.roles.NewRoleResource* attribute), 44, 53
 - role_serializer (*app.blueprints.roles.RoleBaseResource* attribute), 46, 53
 - role_serializer (*app.blueprints.roles.RoleResource* attribute), 48, 54
 - role_serializer (*app.blueprints.roles.RolesSearchResource* attribute), 51, 55
 - RoleBaseResource (*class in app.blueprints.roles*), 45, 53
 - RoleManager (*class in app.managers.role*), 145, 146
 - RoleName (*class in app.serializers.role*), 279, 288
 - RoleResource (*class in app.blueprints.roles*), 47, 53
 - roles (*app.models.role.Role* attribute), 172, 182
 - roles (*app.models.user.User* attribute), 186, 198
 - RoleSeeder (*class in database.seeds.role_seeder*), 374, 375
 - RoleSerializer (*class in app.serializers.role*), 282, 290
 - RoleSerializer.Meta (*class in app.serializers.role*), 290
 - RoleService (*class in app.services.role*), 332, 333
 - RolesSearchResource (*class in app.blueprints.roles*), 50, 54
 - rollback_actions() (*in module database.migrations*), 371, 372
 - root (*app.serializers.core.TimestampField* attribute), 232, 242
 - root (*app.serializers.role.RoleName* attribute), 280, 290
 - root (*app.serializers.user.VerifyRoleId* attribute), 310, 326
 - ROOT_DIRECTORY (*config.Config* attribute), 403, 430
 - ROOT_DIRECTORY (*config.DevConfig* attribute), 411, 431
 - ROOT_DIRECTORY (*config.ProdConfig* attribute), 419, 433
 - ROOT_DIRECTORY (*config.TestConfig* attribute), 426, 435
 - run() (*app.celery.ContextTask* method), 100, 127
 - run_wsgi_app() (*tests.custom_flask_client.CustomFlaskClient* method), 394, 396
 - runner() (*in module tests.conftest*), 390, 391
- ## S
- s() (*app.celery.ContextTask* method), 100, 127
 - save() (*app.managers.base.BaseManager* method), 143
 - save() (*app.managers.document.DocumentManager* method), 144
 - save() (*app.managers.role.RoleManager* method), 146
 - save() (*app.managers.user.UserManager* method), 148
 - save() (*app.models.base.Base* method), 157, 159
 - save() (*app.models.document.Document* method), 168, 170
 - save() (*app.models.role.Role* method), 179, 182
 - save() (*app.models.user.User* method), 194, 198
 - save() (*app.services.base.BaseService* method), 329

- save() (*app.services.document.DocumentService* method), 331
- save() (*app.services.role.RoleService* method), 333
- save() (*app.services.user.UserService* method), 336
- save() (*database.factories.Factory* method), 348, 350, 352
- save() (*database.migrations.aad_create_user_roles_table* method), 358
- save() (*database.migrations.aaf_remove_role_slug_column* method), 361
- save() (*database.migrations.Migration* method), 369, 372
- save_bytes() (*app.utils.file_storage.FileStorage* method), 340
- SearchDocumentResource (class in *app.blueprints.documents*), 37, 41
- SearchSerializer (class in *app.serializers.core*), 225, 234
- SearchSerializer.Meta (class in *app.serializers.core*), 234
- SECRET_KEY (*config.Config* attribute), 404, 430
- SECRET_KEY (*config.DevConfig* attribute), 411, 432
- SECRET_KEY (*config.ProdConfig* attribute), 419, 433
- SECRET_KEY (*config.TestConfig* attribute), 426, 435
- SECURITY_PASSWORD_HASH (*config.Config* attribute), 404, 430
- SECURITY_PASSWORD_HASH (*config.DevConfig* attribute), 411, 432
- SECURITY_PASSWORD_HASH (*config.ProdConfig* attribute), 419, 433
- SECURITY_PASSWORD_HASH (*config.TestConfig* attribute), 426, 435
- SECURITY_PASSWORD_LENGTH_MIN (*config.Config* attribute), 404, 430
- SECURITY_PASSWORD_LENGTH_MIN (*config.DevConfig* attribute), 411, 432
- SECURITY_PASSWORD_LENGTH_MIN (*config.ProdConfig* attribute), 419, 433
- SECURITY_PASSWORD_LENGTH_MIN (*config.TestConfig* attribute), 427, 435
- SECURITY_PASSWORD_SALT (*config.Config* attribute), 404, 430
- SECURITY_PASSWORD_SALT (*config.DevConfig* attribute), 411, 432
- SECURITY_PASSWORD_SALT (*config.ProdConfig* attribute), 419, 433
- SECURITY_PASSWORD_SALT (*config.TestConfig* attribute), 427, 435
- SECURITY_TOKEN_AUTHENTICATION_HEADER (*config.Config* attribute), 404, 430
- SECURITY_TOKEN_AUTHENTICATION_HEADER (*config.DevConfig* attribute), 411, 432
- SECURITY_TOKEN_AUTHENTICATION_HEADER (*config.ProdConfig* attribute), 419, 433
- SECURITY_TOKEN_AUTHENTICATION_HEADER (*config.TestConfig* attribute), 427, 435
- SECURITY_TOKEN_MAX_AGE (*config.Config* attribute), 404, 430
- SECURITY_TOKEN_MAX_AGE (*config.DevConfig* attribute), 411, 432
- SECURITY_TOKEN_MAX_AGE (*config.ProdConfig* attribute), 419, 433
- SECURITY_TOKEN_MAX_AGE (*config.TestConfig* attribute), 427, 435
- seed_actions() (in module *database*), 377
- select() (*app.models.base.Base* class method), 157, 159
- select() (*app.models.document.Document* class method), 168, 170
- select() (*app.models.role.Role* class method), 179, 182
- select() (*app.models.user.User* class method), 194, 198
- select() (*database.migrations.aad_create_user_roles_table._OldUser* class method), 358
- select() (*database.migrations.aaf_remove_role_slug_column._OldRole* class method), 361
- select() (*database.migrations.Migration* class method), 369, 372
- select_queues() (*app.celery.MyCelery* method), 117, 137
- send_create_user_email() (*app.services.task.TaskService* method), 334
- send_email_with_attachments_task() (in module *app.celery.tasks*), 86
- send_event() (*app.celery.ContextTask* method), 100, 127
- send_events (*app.celery.ContextTask* attribute), 91, 127
- send_task() (*app.celery.MyCelery* method), 117, 137
- serialize() (*app.serializers.core.TimestampField* method), 234, 242
- serialize() (*app.serializers.role.RoleName* method), 282, 290
- serialize() (*app.serializers.user.VerifyRoleId* method), 311, 326
- serializer (*app.celery.ContextTask* attribute), 92, 127
- SERVER_NAME (*config.Config* attribute), 404, 430
- SERVER_NAME (*config.DevConfig* attribute), 412, 432
- SERVER_NAME (*config.ProdConfig* attribute), 420, 433
- SERVER_NAME (*config.TestConfig* attribute), 427, 435
- session_transaction() (*tests.custom_flask_client.CustomFlaskClient* method), 394, 396
- set_by_id() (*app.models.base.Base* class method), 157, 159
- set_by_id() (*app.models.document.Document* class method), 168, 170
- set_by_id() (*app.models.role.Role* class method), 179, 182

set_by_id() (*app.models.user.User* class method), 194, 198
 set_by_id() (*database.migrations.aad_create_user_roles_table.OldRole* class method), 358
 set_by_id() (*database.migrations.aaf_remove_role_slug_column.OldRole* class method), 361
 set_by_id() (*database.migrations.Migration* class method), 369, 372
 set_class(*app.serializers.auth.AuthUserConfirmResetPasswordSerializer* property), 202, 219
 set_class(*app.serializers.auth.AuthUserLoginSerializer* property), 208, 225
 set_class(*app.serializers.core._SearchOrderSerializer* property), 248
 set_class(*app.serializers.core._SearchValueSerializer* property), 254
 set_class(*app.serializers.core.SearchSerializer* property), 227, 240
 set_class(*app.serializers.document.DocumentAttachmentSerializer* property), 256, 271
 set_class(*app.serializers.document.DocumentSerializer* property), 262, 278
 set_class(*app.serializers.role.RoleSerializer* property), 284, 296
 set_class(*app.serializers.user.UserExportWordSerializer* property), 298, 317
 set_class(*app.serializers.user.UserSerializer* property), 304, 323
 set_cookie() (*tests.custom_flask_client.CustomFlaskClient* method), 395, 396
 set_current() (*app.celery.MyCelery* method), 118, 137
 set_default() (*app.celery.MyCelery* method), 118, 137
 setup_security() (*app.celery.MyCelery* method), 118, 137
 shadow_name() (*app.celery.ContextTask* method), 101, 128
 si() (*app.celery.ContextTask* method), 101, 128
 signature() (*app.celery.ContextTask* method), 101, 128
 signature() (*app.celery.MyCelery* method), 118, 137
 signature_from_request() (*app.celery.ContextTask* method), 101, 128
 size (*app.models.document.Document* attribute), 162, 170
 slug (*database.migrations.aaf_remove_role_slug_column.OldRole* attribute), 361
 sluglify_name() (*app.serializers.role.RoleSerializer* method), 287, 296
 soft_time_limit (*app.celery.ContextTask* attribute), 92, 128
 starmap() (*app.celery.ContextTask* method), 102, 128
 start() (*app.celery.MyCelery* method), 118, 137
 start_strategy() (*app.celery.ContextTask* method), 102, 128
 steps (*app.celery.MyCelery* attribute), 109, 138
 STORAGE_DIRECTORY (*config.Config* attribute), 404, 430
 STORAGE_DIRECTORY (*config.DevConfig* attribute), 412, 434
 STORAGE_DIRECTORY (*config.ProdConfig* attribute), 420, 434
 STORAGE_DIRECTORY (*config.TestConfig* attribute), 427, 435
 store_eager_result (*app.celery.ContextTask* attribute), 92, 128
 store_errors_even_if_ignored (*app.celery.ContextTask* attribute), 92, 128
 Strategy (*app.celery.ContextTask* attribute), 89, 120
 strerror (*app.exceptions.FileEmptyError* attribute), 140
 subclass_with_self() (*app.celery.MyCelery* method), 119, 138
 subtask() (*app.celery.ContextTask* method), 102, 128
 subtask_from_request() (*app.celery.ContextTask* method), 102, 128
 SWAGGER_API_URL (*config.Config* attribute), 404, 430
 SWAGGER_API_URL (*config.DevConfig* attribute), 412, 432
 SWAGGER_API_URL (*config.ProdConfig* attribute), 420, 434
 SWAGGER_API_URL (*config.TestConfig* attribute), 427, 435
 SWAGGER_URL (*config.Config* attribute), 404, 430
 SWAGGER_URL (*config.DevConfig* attribute), 412, 432
 SWAGGER_URL (*config.ProdConfig* attribute), 420, 434
 SWAGGER_URL (*config.TestConfig* attribute), 427, 435
 SYSTEM (*app.celery.MyCelery* attribute), 105, 130

T

table_exists() (*app.models.base.Base* class method), 158, 159
 table_exists() (*app.models.document.Document* class method), 168, 170
 table_exists() (*app.models.role.Role* class method), 179, 182
 table_exists() (*app.models.user.User* class method), 194, 198
 table_exists() (*database.migrations.aad_create_user_roles_table.OldRole* class method), 358
 table_exists() (*database.migrations.aaf_remove_role_slug_column.OldRole* class method), 361
 table_exists() (*database.migrations.Migration* class method), 370, 372
 Task (*app.celery.MyCelery* attribute), 105, 130
 task() (*app.celery.MyCelery* method), 119, 138
 task_always_eager (*config.Config* attribute), 406, 431
 task_always_eager (*config.DevConfig* attribute), 413, 432

- `task_always_eager` (*config.ProdConfig* attribute), 421, 434
- `task_always_eager` (*config.TestConfig* attribute), 428, 435
- `task_cls` (*app.celery.MyCelery* attribute), 109, 138
- `task_default_rate_limit` (*config.Config* attribute), 406, 431
- `task_default_rate_limit` (*config.DevConfig* attribute), 413, 432
- `task_default_rate_limit` (*config.ProdConfig* attribute), 421, 434
- `task_default_rate_limit` (*config.TestConfig* attribute), 429, 436
- `task_serializer` (*config.Config* attribute), 406, 431
- `task_serializer` (*config.DevConfig* attribute), 413, 432
- `task_serializer` (*config.ProdConfig* attribute), 421, 434
- `task_serializer` (*config.TestConfig* attribute), 429, 436
- `task_service` (*app.blueprints.tasks.TaskResource* attribute), 56, 60
- `task_service` (*app.blueprints.tasks.TaskStatusResource* attribute), 58, 60
- `task_service` (*app.blueprints.users.ExportUsersExcelAndWordResource* attribute), 62, 80
- `task_service` (*app.blueprints.users.ExportUsersExcelResource* attribute), 65, 80
- `task_service` (*app.blueprints.users.ExportUsersWordResource* attribute), 68, 81
- `task_service` (*app.blueprints.users.NewUserResource* attribute), 70, 82
- `task_service` (*app.blueprints.users.UserBaseResource* attribute), 73, 82
- `task_service` (*app.blueprints.users.UserResource* attribute), 75, 83
- `task_service` (*app.blueprints.users.UsersSearchResource* attribute), 78, 84
- `task_track_started` (*config.Config* attribute), 406, 431
- `task_track_started` (*config.DevConfig* attribute), 413, 432
- `task_track_started` (*config.ProdConfig* attribute), 421, 434
- `task_track_started` (*config.TestConfig* attribute), 429, 436
- `TaskResource` (class in *app.blueprints.tasks*), 55, 59
- `tasks` (*app.celery.MyCelery* attribute), 109, 138
- `TaskService` (class in *app.services.task*), 333, 334
- `TaskStatusResource` (class in *app.blueprints.tasks*), 57, 60
- `test_api_middleware()` (in module *tests.test_middleware*), 398
- `test_check_task_status()` (in module *tests.blueprints.test_tasks*), 384
- `test_config()` (in module *tests.test_config*), 397
- `test_create_invalid_user_endpoint()` (in module *tests.blueprints.test_users*), 385, 386
- `test_create_user_email_task()` (in module *tests.celery.test_tasks*), 389
- `test_create_user_endpoint()` (in module *tests.blueprints.test_users*), 385, 386
- `test_create_word_and_excel_documents()` (in module *tests.celery.test_tasks*), 389
- `test_delete_document()` (in module *tests.blueprints.test_documents*), 381, 382
- `test_delete_role_endpoint()` (in module *tests.blueprints.test_roles*), 383, 384
- `test_delete_user_endpoint()` (in module *tests.blueprints.test_users*), 385, 386
- `test_export_excel_and_word_endpoint()` (in module *tests.blueprints.test_users*), 386
- `test_export_excel_endpoint()` (in module *tests.blueprints.test_users*), 386, 387
- `test_export_excel_task()` (in module *tests.celery.test_excel*), 388
- `test_export_word_endpoint()` (in module *tests.blueprints.test_users*), 386, 387
- `test_export_word_task()` (in module *tests.celery.test_word*), 389
- `test_get_close_db()` (in module *tests.test_db*), 397
- `test_get_document_data()` (in module *tests.blueprints.test_documents*), 382
- `test_get_document_file()` (in module *tests.blueprints.test_documents*), 382
- `test_get_role_endpoint()` (in module *tests.blueprints.test_roles*), 383, 384
- `test_get_user_endpoint()` (in module *tests.blueprints.test_users*), 386, 387
- `test_mail_record_messages()` (in module *tests.test_mail*), 398
- `test_no_api_middleware()` (in module *tests.test_middleware*), 398
- `test_register_context_task()` (in module *tests.celery.test_celery*), 388
- `test_request_reset_password()` (in module *tests.blueprints.test_auth*), 380
- `test_reset_password()` (in module *tests.blueprints.test_auth*), 380
- `test_reset_password_email_task()` (in module *tests.celery.test_tasks*), 389
- `test_save_document()` (in module *tests.blueprints.test_documents*), 382
- `test_save_role_endpoint()` (in module *tests.blueprints.test_roles*), 383, 384
- `test_search_document()` (in module *tests.blueprints.test_documents*), 382
- `test_search_roles_endpoint()` (in module *tests.blueprints.test_roles*), 383, 384

tests.blueprints.test_roles), 383, 384
test_search_users_endpoint() (in module *tests.blueprints.test_users*), 386, 387
test_send_email_with_attachments_task() (in module *tests.celery.test_tasks*), 389
test_update_document() (in module *tests.blueprints.test_documents*), 382
test_update_role_endpoint() (in module *tests.blueprints.test_roles*), 384
test_update_user_endpoint() (in module *tests.blueprints.test_users*), 386, 387
TEST_USER_EMAIL (*config.Config* attribute), 405, 430
TEST_USER_EMAIL (*config.DevConfig* attribute), 412, 432
TEST_USER_EMAIL (*config.ProdConfig* attribute), 420, 434
TEST_USER_EMAIL (*config.TestConfig* attribute), 427, 435
test_user_login() (in module *tests.blueprints.test_auth*), 380
test_user_logout() (in module *tests.blueprints.test_auth*), 380
TEST_USER_PASSWORD (*config.Config* attribute), 405, 430
TEST_USER_PASSWORD (*config.DevConfig* attribute), 412, 432
TEST_USER_PASSWORD (*config.ProdConfig* attribute), 420, 434
TEST_USER_PASSWORD (*config.TestConfig* attribute), 428, 435
test_validate_reset_password() (in module *tests.blueprints.test_auth*), 380
test_welcome_api() (in module *tests.blueprints.test_base*), 381
TestConfig (class in *config*), 422, 434
TESTING (*config.Config* attribute), 405, 430
TESTING (*config.DevConfig* attribute), 412, 432
TESTING (*config.ProdConfig* attribute), 420, 434
TESTING (*config.TestConfig* attribute), 427, 435
tests
 module, 377
tests.blueprints
 module, 379
tests.blueprints.test_auth
 module, 379
tests.blueprints.test_base
 module, 380
tests.blueprints.test_documents
 module, 381
tests.blueprints.test_roles
 module, 383
tests.blueprints.test_tasks
 module, 384
tests.blueprints.test_users
 module, 385
tests.celery
 module, 387
tests.celery.test_celery
 module, 387
tests.celery.test_excel
 module, 388
tests.celery.test_tasks
 module, 388
tests.celery.test_word
 module, 389
tests.conftest
 module, 390
tests.custom_flask_client
 module, 391
tests.test_config
 module, 397
tests.test_db
 module, 397
tests.test_mail
 module, 397
tests.test_middleware
 module, 398
tf_send_security_token() (*app.models.user.User* method), 194, 199
thread_oid (*app.celery.MyCelery* property), 109, 139
throws (*app.celery.ContextTask* attribute), 92, 129
time_limit (*app.celery.ContextTask* attribute), 92, 129
TimestampField (class in *app.serializers.core*), 231, 240
timezone (*app.celery.MyCelery* attribute), 109, 139
timezone (*config.Config* attribute), 406, 431
timezone (*config.DevConfig* attribute), 413, 432
timezone (*config.ProdConfig* attribute), 421, 434
timezone (*config.TestConfig* attribute), 429, 436
to_readable() (in module *app.utils*), 346
token_required() (in module *app.utils.decorators*), 338
trace() (*tests.custom_flask_client.CustomFlaskClient* method), 395, 396
track_started (*app.celery.ContextTask* attribute), 92, 129
trail (*app.celery.ContextTask* attribute), 93, 129
truncate_table() (*app.models.base.Base* class method), 158, 159
truncate_table() (*app.models.document.Document* class method), 168, 170
truncate_table() (*app.models.role.Role* class method), 179, 182
truncate_table() (*app.models.user.User* class method), 194, 199
truncate_table() (*database.migrations.aad_create_user_roles_table._C* class method), 358

`truncate_table()` (`database.migrations.aaf_remove_role_slug_column._OldRole` class method), 362
`truncate_table()` (`database.migrations.Migration` class method), 370, 372
`TYPE_MAPPING` (`app.serializers.auth.AuthUserConfirmResetSerializer` attribute), 202, 214
`TYPE_MAPPING` (`app.serializers.auth.AuthUserLoginSerializer` attribute), 208, 220
`TYPE_MAPPING` (`app.serializers.core._SearchOrderSerializer` attribute), 243
`TYPE_MAPPING` (`app.serializers.core._SearchValueSerializer` attribute), 249
`TYPE_MAPPING` (`app.serializers.core.SearchSerializer` attribute), 226, 235
`TYPE_MAPPING` (`app.serializers.document.DocumentAttachmentSerializer` attribute), 255, 267
`TYPE_MAPPING` (`app.serializers.document.DocumentSerializer` attribute), 261, 272
`TYPE_MAPPING` (`app.serializers.role.RoleSerializer` attribute), 283, 290
`TYPE_MAPPING` (`app.serializers.user.UserExportWordSerializer` attribute), 298, 312
`TYPE_MAPPING` (`app.serializers.user.UserSerializer` attribute), 303, 317
`typing` (`app.celery.ContextTask` attribute), 93, 129

U

`unwrap()` (`app.models.base.Base` method), 158, 159
`unwrap()` (`app.models.document.Document` method), 168, 170
`unwrap()` (`app.models.role.Role` method), 180, 182
`unwrap()` (`app.models.user.User` method), 194, 199
`unwrap()` (`database.migrations.aad_create_user_roles_table._OldUser` method), 358
`unwrap()` (`database.migrations.aaf_remove_role_slug_column._OldRole` method), 362
`unwrap()` (`database.migrations.Migration` method), 370, 372
`up()` (`database.migrations.aaa_add_genre_column_on_user_table.AddGenreColumnOnUserTable` method), 353
`up()` (`database.migrations.aab_add_created_by_column_on_user_table.AddCreatedByColumnOnUserTable` method), 354
`up()` (`database.migrations.aac_create_documents_table.CreateDocumentsTable` method), 355
`up()` (`database.migrations.aad_create_user_roles_table.CreateUserRolesTable` method), 356
`up()` (`database.migrations.aaf_remove_role_slug_column.RemoveRoleSlugColumn` method), 359, 360
`up()` (`database.migrations.aag_add_fs_uniquifier_column_on_users_table.AddFsUniquifierColumnOnUsersTable` method), 363
`update()` (`app.models.base.Base` class method), 158, 159
`update()` (`app.models.document.Document` class method), 168, 170
`update()` (`app.models.role.Role` class method), 180, 182
`update()` (`app.models.user.User` class method), 195, 199
`update()` (`database.migrations.aad_create_user_roles_table._OldUser` class method), 358
`update()` (`database.migrations.aaf_remove_role_slug_column._OldRole` class method), 362
`update()` (`database.migrations.Migration` class method), 370, 372
`update_state()` (`app.celery.ContextTask` method), 102, 129
`updated_at` (`app.models.document.Document` attribute), 162, 170
`updated_at` (`app.models.role.Role` attribute), 172, 182
`updated_at` (`app.models.user.User` attribute), 186, 199
`updated_at` (`database.migrations.aad_create_user_roles_table._OldUser` attribute), 358
`updated_at` (`database.migrations.aaf_remove_role_slug_column._OldRole` attribute), 362
`url` (`app.models.document.Document` property), 162, 170
`us_send_security_token()` (`app.models.user.User` method), 195, 199
`User` (class in `app.models.user`), 182, 196
`user_options` (`app.celery.MyCelery` attribute), 109, 139
`user_serializer` (`app.blueprints.users.ExportUsersExcelAndWordResource` attribute), 62, 80
`user_serializer` (`app.blueprints.users.ExportUsersExcelResource` attribute), 65, 80
`user_serializer` (`app.blueprints.users.ExportUsersWordResource` attribute), 68, 81
`user_serializer` (`app.blueprints.users.NewUserResource` attribute), 71, 82
`user_serializer` (`app.blueprints.users.UserBaseResource` attribute), 73, 82
`user_serializer` (`app.blueprints.users.UserResource` attribute), 75, 83
`user_serializer` (`app.blueprints.users.UsersSearchResource` attribute), 78, 84
`user_service` (`app.blueprints.users.ExportUsersExcelAndWordResource` attribute), 63, 80
`user_service` (`app.blueprints.users.ExportUsersExcelResource` attribute), 66, 80
`user_service` (`app.blueprints.users.ExportUsersWordResource` attribute), 68, 81
`user_service` (`app.blueprints.users.NewUserResource` attribute), 71, 82
`user_service` (`app.blueprints.users.UserBaseResource` attribute), 73, 82
`user_service` (`app.blueprints.users.UserResource` attribute), 75, 83
`user_service` (`app.blueprints.users.UsersSearchResource` attribute), 78, 84
`UserBaseResource` (class in `app.blueprints.users`), 72, 82

UserExportWordSerializer (class in *app.serializers.user*), 297, 311
 UserExportWordSerializer.Meta (class in *app.serializers.user*), 311
 UserManager (class in *app.managers.user*), 147, 148
 UserResource (class in *app.blueprints.users*), 74, 83
 userrolethrough_set (*app.models.role.Role* attribute), 173, 182
 userrolethrough_set (*app.models.user.User* attribute), 186, 199
 users (*app.models.role.Role* attribute), 173, 182
 UserSeeder (class in *database.seeds.user_seeder*), 375, 376
 UserSerializer (class in *app.serializers.user*), 303, 317
 UserSerializer.Meta (class in *app.serializers.user*), 317
 UserService (class in *app.services.user*), 335, 336
 UsersSearchResource (class in *app.blueprints.users*), 77, 83
 uses_utc_timezone() (*app.celery.MyCelery* method), 120, 139

V

valid_request_file()
 (*app.serializers.document.DocumentSerializer* static method), 265, 278
 validate() (*app.serializers.auth.AuthUserConfirmResetPasswordSerializer* method), 207, 219
 validate() (*app.serializers.auth.AuthUserLoginSerializer* method), 212, 225
 validate() (*app.serializers.core._SearchOrderSerializer* method), 248
 validate() (*app.serializers.core._SearchValueSerializer* method), 254
 validate() (*app.serializers.core.SearchSerializer* method), 230, 240
 validate() (*app.serializers.document.DocumentAttachmentSerializer* method), 260, 271
 validate() (*app.serializers.document.DocumentSerializer* method), 266, 278
 validate() (*app.serializers.role.RoleSerializer* method), 288, 296
 validate() (*app.serializers.user.UserExportWordSerializer* method), 302, 317
 validate() (*app.serializers.user.UserSerializer* method), 308, 323
 validate_email() (*app.serializers.auth.AuthUserLoginSerializer* method), 213, 225
 validate_email() (*app.serializers.user.UserSerializer* method), 308, 324
 validate_id() (*app.serializers.document.DocumentSerializer* method), 266, 279
 validate_id() (*app.serializers.role.RoleSerializer* method), 288, 296
 validate_id() (*app.serializers.user.UserSerializer* method), 308, 324
 validate_model() (*app.models.base.Base* class method), 158, 159
 validate_model() (*app.models.document.Document* class method), 169, 170
 validate_model() (*app.models.role.Role* class method), 180, 182
 validate_model() (*app.models.user.User* class method), 195, 199
 validate_model() (*database.migrations.aad_create_user_roles_table.C* class method), 359
 validate_model() (*database.migrations.aaf_remove_role_slug_column.C* class method), 362
 validate_model() (*database.migrations.Migration* class method), 370, 372
 validate_name() (*app.serializers.role.RoleSerializer* method), 288, 296
 validate_password()
 (*app.serializers.auth.AuthUserLoginSerializer* method), 213, 225
 validate_payload() (*app.blueprints.auth.AuthBaseResource* method), 11, 21
 validate_payload() (*app.blueprints.auth.AuthUserLoginResource* method), 13, 21
 validate_payload() (*app.blueprints.auth.AuthUserLogoutResource* method), 15, 22
 validate_payload() (*app.blueprints.auth.RequestResetPasswordResource* method), 18, 23
 validate_payload() (*app.blueprints.auth.ResetPasswordResource* method), 20, 23
 validate_payload() (*app.blueprints.base.BaseResource* method), 25, 28
 validate_payload() (*app.blueprints.base.WelcomeResource* method), 28, 29
 validate_payload() (*app.blueprints.documents.DocumentBaseResource* method), 31, 40
 validate_payload() (*app.blueprints.documents.DocumentResource* method), 34, 40
 validate_payload() (*app.blueprints.documents.NewDocumentResource* method), 37, 41
 validate_payload() (*app.blueprints.documents.SearchDocumentResource* method), 39, 42
 validate_payload() (*app.blueprints.roles.NewRoleResource* method), 45, 53
 validate_payload() (*app.blueprints.roles.RoleBaseResource* method), 47, 53
 validate_payload() (*app.blueprints.roles.RoleResource* method), 50, 54
 validate_payload() (*app.blueprints.roles.RolesSearchResource* method), 52, 55
 validate_payload() (*app.blueprints.tasks.TaskResource* method), 55, 56

method), 57, 60
validate_payload() (*app.blueprints.tasks.TaskStatusResource* *tribute*), 422, 434
method), 59, 60
validate_payload() (*app.blueprints.users.ExportUsersExcelAndWordResource*), 422, 436
method), 64, 80
validate_payload() (*app.blueprints.users.ExportUsersExcelResource*), 266, 279
method), 67, 80
validate_payload() (*app.blueprints.users.ExportUsersWordResource*
method), 69, 81
validate_payload() (*app.blueprints.users.NewUserResource*
method), 72, 82
validate_payload() (*app.blueprints.users.UserBaseResource*
method), 74, 82
validate_payload() (*app.blueprints.users.UserResource*
method), 77, 83
validate_payload() (*app.blueprints.users.UsersSearchResource*
method), 79, 84
validate_token() (*app.serializers.auth.AuthUserConfirmResetPasswordSerializer*
method), 207, 219
verify_and_update_password()
(*app.models.user.User* *method*), 195, 199
verify_auth_token() (*app.models.user.User* *method*),
195, 199
verify_reset_token() (*app.models.user.User* *static*
method), 196, 199
VerifyRoleId (*class in app.serializers.user*), 308, 324

W

WelcomeResource (*class in app.blueprints.base*), 26, 28
with_traceback() (*app.celery.ContextTask.MaxRetriesExceededError*
method), 120
with_traceback() (*app.celery.ContextTask.OperationalError*
method), 120
with_traceback() (*app.exceptions.FileEmptyError*
method), 140
with_traceback() (*app.utils.libreoffice.LibreOfficeError*
method), 341
WorkController (*app.celery.MyCelery* *attribute*), 105,
130
Worker (*app.celery.MyCelery* *attribute*), 105, 130
worker_log_format (*config.Config* *attribute*), 406, 431
worker_log_format (*config.DevConfig* *attribute*), 414,
432
worker_log_format (*config.ProdConfig* *attribute*), 422,
434
worker_log_format (*config.TestConfig* *attribute*), 429,
436
worker_main() (*app.celery.MyCelery* *method*), 120,
139
worker_task_log_format (*config.Config* *attribute*),
406, 431
worker_task_log_format (*config.DevConfig* *at-*
tribute), 414, 432