flask_api

Rubén Rodríguez Ramírez

Jan 05, 2021

CONTENTS

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.

CHAPTER

ONE

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 appropiate 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 filles 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 appropiate 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 appropiate 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.

CHAPTER

TWO

SKELETON APP STRUCTURE

The project structure looks like this:

```
flask_api
├── /app
│   ├── /blueprints
│   │   └── ...
│   ├── /celery
│   │   └── ...
│   ├── /models
│   │   └── ...
│   ├── /templates
│   │   └── ...
│   ├── /utils
│   │   └── ...
│   ├── __init__.py
│   ├── extensions.py
│   └── middleware.py
├── /database
│   ├── factories
│   │   └── ...
│   ├── migrations
│   │   └── ...
│   ├── seeds
│   │   └── ...
│   └── __init__.py
├── /log
│   └── ...
├── /storage
│   └── ...
├── /tests
│   └── ...
├── config.py
├── manage.py
└── requirements.txt
```

app	Package for building a Flask application.
database	Package for managing the database.
tests	Package for testing the application.
config	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

app.blueprints	Registers Flask blueprints.
app.celery	Registers Celery tasks.
app.extensions	Registers third party extensions.
app.middleware	WSGI middleware for validating requests content type.
app.models	Registers database models.
app.utils	Collection of functions and classes which make common patterns shorter and easier.

2.1.1 app.blueprints

Description

Registers Flask blueprints.

Modules

app.blueprints.auth
app.blueprints.base
app.blueprints.documents
app.blueprints.roles
app.blueprints.tasks
app.blueprints.users

app.blueprints.auth

Description

Classes

AuthUserLoginResource([api])
AuthUserLogoutResource([api])
RequestResetPasswordResource([api])
ResetPasswordResource([api])

app.blueprints.auth.AuthUserLoginResource

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

Attributes

AuthUserLoginResource.decorators
AuthUserLoginResource.method_decorators
AuthUserLoginResource.methods
AuthUserLoginResource.provide_automatic_options
AuthUserLoginResource.representations

app.blueprints.auth.AuthUserLoginResource.decorators

AuthUserLoginResource.**decorators = ()**

app.blueprints.auth.AuthUserLoginResource.method_decorators

AuthUserLoginResource.**method_decorators = []**

app.blueprints.auth.AuthUserLoginResource.methods

AuthUserLoginResource.**methods = {'POST'}**

app.blueprints.auth.AuthUserLoginResource.provide_automatic_options

AuthUserLoginResource.**provide_automatic_options = None**

app.blueprints.auth.AuthUserLoginResource.representations

AuthUserLoginResource.**representations = None**

Methods

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

app.blueprints.auth.AuthUserLoginResource.__init__

AuthUserLoginResource.**__init__**(*api=None*, **args*, ***kwargs*)
Initialize self. See help(type(self)) for accurate signature.

app.blueprints.auth.AuthUserLoginResource.as_view

classmethod AuthUserLoginResource.**as_view**(*name*, **class_args*, ***class_kwargs*)
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: flask_restx.resource.Resource

Attributes

AuthUserLogoutResource.decorators
AuthUserLogoutResource.method_decorators
AuthUserLogoutResource.methods
AuthUserLogoutResource.provide_automatic_options
AuthUserLogoutResource.representations

app.blueprints.auth.AuthUserLogoutResource.decorators

AuthUserLogoutResource.**decorators = ()**

app.blueprints.auth.AuthUserLogoutResource.method_decorators

AuthUserLogoutResource.**method_decorators = []**

app.blueprints.auth.AuthUserLogoutResource.methods

AuthUserLogoutResource.**methods = {'POST'}**

app.blueprints.auth.AuthUserLogoutResource.provide_automatic_options

AuthUserLogoutResource.**provide_automatic_options = None**

app.blueprints.auth.AuthUserLogoutResource.representations

AuthUserLogoutResource.**representations = None**

Methods

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

app.blueprints.auth.AuthUserLogoutResource.__init__

AuthUserLogoutResource.**__init__**(*api=None*, **args*, ***kwargs*)
Initialize self. See help(type(self)) for accurate signature.

app.blueprints.auth.AuthUserLogoutResource.as_view

classmethod AuthUserLogoutResource.**as_view**(*name*, **class_args*, ***class_kwargs*)
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: flask_restx.resource.Resource

Attributes

RequestResetPasswordResource.decorators
RequestResetPasswordResource.method_decorators
RequestResetPasswordResource.methods
RequestResetPasswordResource.provide_automatic_options
RequestResetPasswordResource.representations

app.blueprints.auth.RequestResetPasswordResource.decorators

RequestResetPasswordResource.**decorators = ()**

app.blueprints.auth.RequestResetPasswordResource.method_decorators

RequestResetPasswordResource.**method_decorators = []**

app.blueprints.auth.RequestResetPasswordResource.methods

RequestResetPasswordResource.**methods = {'POST'}**

app.blueprints.auth.RequestResetPasswordResource.provide_automatic_options

RequestResetPasswordResource.**provide_automatic_options = None**

app.blueprints.auth.RequestResetPasswordResource.representations

RequestResetPasswordResource.**representations = None**

Methods

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

app.blueprints.auth.RequestResetPasswordResource.__init__

RequestResetPasswordResource.**__init__**(*api=None*, **args*, ***kwargs*)
Initialize self. See help(type(self)) for accurate signature.

app.blueprints.auth.RequestResetPasswordResource.as_view

classmethod RequestResetPasswordResource.**as_view**(*name*, **class_args*, ***class_kwargs*)
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: flask_restx.resource.Resource

Attributes

ResetPasswordResource.decorators
ResetPasswordResource.method_decorators
ResetPasswordResource.methods
ResetPasswordResource.provide_automatic_options
ResetPasswordResource.representations

app.blueprints.auth.ResetPasswordResource.decorators

ResetPasswordResource.**decorators = ()**

app.blueprints.auth.ResetPasswordResource.method_decorators

ResetPasswordResource.**method_decorators = []**

app.blueprints.auth.ResetPasswordResource.methods

ResetPasswordResource.**methods = {'GET', 'POST'}**

app.blueprints.auth.ResetPasswordResource.provide_automatic_options

ResetPasswordResource.**provide_automatic_options = None**

app.blueprints.auth.ResetPasswordResource.representations

ResetPasswordResource.**representations = None**

Methods

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

app.blueprints.auth.ResetPasswordResource.__init__

ResetPasswordResource.**__init__**(*api=None*, **args*, ***kwargs*)
Initialize self. See help(type(self)) for accurate signature.

app.blueprints.auth.ResetPasswordResource.as_view

classmethod ResetPasswordResource.**as_view**(*name*, **class_args*, ***class_kwargs*)
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.**AuthUserLoginResource**(*api=None*, **args*, ***kwargs*)

classmethod as_view(*name*, **class_args*, ***class_kwargs*)
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 = ()

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 = {'POST'}

post() → tuple

provide_automatic_options = None

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*, **class_args*, ***class_kwargs*)
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 = ()

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 = {'POST'}

post() → tuple

provide_automatic_options = None

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*, **class_args*, ***class_kwargs*)
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 = ()

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 = {'POST'}

post() → tuple

provide_automatic_options = None

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*, **class_args*, ***class_kwargs*)
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 = ()

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 = {'GET', 'POST'}

post(*token: str*) → tuple

provide_automatic_options = None

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

BaseResource.decorators	
BaseResource.method_decorators	
BaseResource.methods	
BaseResource.provide_automatic_options	
BaseResource.representations	

app.blueprints.base.BaseResource.decorators

BaseResource.**decorators = ()**

app.blueprints.base.BaseResource.method_decorators

BaseResource.**method_decorators = []**

app.blueprints.base.BaseResource.methods

BaseResource.**methods = None**

app.blueprints.base.BaseResource.provide_automatic_options

BaseResource.**provide_automatic_options = None**

app.blueprints.base.BaseResource.representations

BaseResource.**representations = None**

Methods

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

app.blueprints.base.BaseResource.__init__

BaseResource.**__init__**(*api=None*, **args*, ***kwargs*)
Initialize self. See help(type(self)) for accurate signature.

app.blueprints.base.BaseResource.as_view

classmethod BaseResource.**as_view**(*name*, **class_args*, ***class_kwargs*)
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.create_search_query

BaseResource.**create_search_query**(*query: peewee.ModelSelect*, *request_data: dict*) → peewee.ModelSelect

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.get_request_query_fields

BaseResource.**get_request_query_fields**(*request_data: dict*) → tuple

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

WelcomeResource.decorators
WelcomeResource.method_decorators
WelcomeResource.methods
WelcomeResource.provide_automatic_options
WelcomeResource.representations

app.blueprints.base.WelcomeResource.decorators

WelcomeResource.**decorators = ()**

app.blueprints.base.WelcomeResource.method_decorators

WelcomeResource.**method_decorators = []**

app.blueprints.base.WelcomeResource.methods

WelcomeResource.**methods = {'GET'}**

app.blueprints.base.WelcomeResource.provide_automatic_options

WelcomeResource.**provide_automatic_options = None**

app.blueprints.base.WelcomeResource.representations

WelcomeResource.**representations = None**

Methods

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

app.blueprints.base.WelcomeResource.__init__

WelcomeResource.**__init__**(*api=None*, **args*, ***kwargs*)
Initialize self. See help(type(self)) for accurate signature.

app.blueprints.base.WelcomeResource.as_view

classmethod WelcomeResource.**as_view**(*name*, **class_args*, ***class_kwargs*)
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*, **class_args*, ***class_kwargs*)
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.

create_search_query(*query: peewee.ModelSelect*, *request_data: dict*) → peewee.ModelSelect

db_model
alias of playhouse.flask_utils.FlaskDB.get_model_class.<locals>.BaseModel

decorators = ()

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_request_query_fields(*request_data: dict*) → tuple

method_decorators = []

methods = None

provide_automatic_options = None

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*, **class_args*, ***class_kwargs*)
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 = ()

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 = {'GET'}

provide_automatic_options = None

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
DocumentBaseResource. document_serializer
DocumentBaseResource. method_decorators
DocumentBaseResource.methods
DocumentBaseResource. provide_automatic_options
DocumentBaseResource. representations
DocumentBaseResource. request_field_name

app.blueprints.documents.DocumentBaseResource.decorators

DocumentBaseResource.**decorators** = ()

app.blueprints.documents.DocumentBaseResource.document_serializer

DocumentBaseResource.**document_serializer = <DocumentSchema(many=False)>**

app.blueprints.documents.DocumentBaseResource.method_decorators

DocumentBaseResource.**method_decorators = []**

app.blueprints.documents.DocumentBaseResource.methods

DocumentBaseResource.**methods = None**

app.blueprints.documents.DocumentBaseResource.provide_automatic_options

DocumentBaseResource.**provide_automatic_options = None**

app.blueprints.documents.DocumentBaseResource.representations

DocumentBaseResource.**representations = None**

app.blueprints.documents.DocumentBaseResource.request_field_name

DocumentBaseResource.**request_field_name = 'document'**

Methods

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

app.blueprints.documents.DocumentBaseResource.__init__

DocumentBaseResource.**__init__**(*api=None*, **args*, ***kwargs*)
Initialize self. See help(type(self)) for accurate signature.

app.blueprints.documents.DocumentBaseResource.as_view

classmethod DocumentBaseResource.**as_view**(*name*, **class_args*, ***class_kwargs*)
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.create_search_query

DocumentBaseResource.**create_search_query**(*query: peewee.ModelSelect*, *request_data: dict*) → peewee.ModelSelect

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.get_document_content

static DocumentBaseResource.**get_document_content**(*document_id: int*)

app.blueprints.documents.DocumentBaseResource.get_document_data

DocumentBaseResource.**get_document_data**(*document_id: int*) → tuple

app.blueprints.documents.DocumentBaseResource.get_request_file

DocumentBaseResource.**get_request_file**() → dict

app.blueprints.documents.DocumentBaseResource.get_request_query_fields

DocumentBaseResource.**get_request_query_fields**(*request_data: dict*) → tuple

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

DocumentResource.decorators
DocumentResource.document_serializer
DocumentResource.method_decorators
DocumentResource.methods
DocumentResource.provide_automatic_options
DocumentResource.representations
DocumentResource.request_field_name

app.blueprints.documents.DocumentResource.decorators

DocumentResource.**decorators = ()**

app.blueprints.documents.DocumentResource.document_serializer

DocumentResource.**document_serializer = <DocumentSchema(many=False)>**

app.blueprints.documents.DocumentResource.method_decorators

DocumentResource.**method_decorators = []**

app.blueprints.documents.DocumentResource.methods

DocumentResource.**methods = {'DELETE', 'GET', 'PUT'}**

app.blueprints.documents.DocumentResource.provide_automatic_options

DocumentResource.**provide_automatic_options = None**

app.blueprints.documents.DocumentResource.representations

DocumentResource.**representations = None**

app.blueprints.documents.DocumentResource.request_field_name

DocumentResource.**request_field_name = 'document'**

Methods

DocumentResource.__init__([api])	Initialize self.
DocumentResource.as_view(name, *class_args, ...)	Converts the class into an actual view function that can be used with the routing system.
DocumentResource.create_search_query(query, ...)	
DocumentResource.delete(document_id)	
DocumentResource.dispatch_request(*args, ...)	Subclasses have to override this method to implement the actual view function code.
DocumentResource.get(document_id)	
DocumentResource.get_document_content(...)	
DocumentResource.get_document_data(document_id)	
DocumentResource.get_request_file()	
DocumentResource.get_request_query_fields(...)	
DocumentResource.put(document_id)	

continues on next page

Table 22 – continued from previous page

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

app.blueprints.documents.DocumentResource.__init__

DocumentResource.**__init__**(*api=None*, **args*, ***kwargs*)
Initialize self. See help(type(self)) for accurate signature.

app.blueprints.documents.DocumentResource.as_view

classmethod DocumentResource.**as_view**(*name*, **class_args*, ***class_kwargs*)
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.create_search_query

DocumentResource.**create_search_query**(*query: peewee.ModelSelect*, *request_data: dict*) → peewee.ModelSelect

app.blueprints.documents.DocumentResource.delete

DocumentResource.**delete**(*document_id: int*)

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.get_document_content

static DocumentResource.**get_document_content**(*document_id: int*)

app.blueprints.documents.DocumentResource.get_document_data

DocumentResource.**get_document_data**(*document_id: int*) → tuple

app.blueprints.documents.DocumentResource.get_request_file

DocumentResource.**get_request_file**() → dict

app.blueprints.documents.DocumentResource.get_request_query_fields

DocumentResource.**get_request_query_fields**(*request_data: dict*) → 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

NewDocumentResource.decorators
NewDocumentResource.document_serializer
NewDocumentResource.method_decorators
NewDocumentResource.methods
NewDocumentResource.parser
NewDocumentResource.provide_automatic_options
NewDocumentResource.representations
NewDocumentResource.request_field_name

app.blueprints.documents.NewDocumentResource.decorators

NewDocumentResource.**decorators = ()**

app.blueprints.documents.NewDocumentResource.document_serializer

NewDocumentResource.**document_serializer = <DocumentSchema(many=False)>**

app.blueprints.documents.NewDocumentResource.method_decorators

NewDocumentResource.**method_decorators = []**

app.blueprints.documents.NewDocumentResource.methods

NewDocumentResource.**methods = {'POST'}**

app.blueprints.documents.NewDocumentResource.parser

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

app.blueprints.documents.NewDocumentResource.provide_automatic_options

NewDocumentResource.**provide_automatic_options = None**

app.blueprints.documents.NewDocumentResource.representations

NewDocumentResource.**representations = None**

app.blueprints.documents.NewDocumentResource.request_field_name

NewDocumentResource.**request_field_name = 'document'**

Methods

NewDocumentResource.__init__([api])	Initialize self.
NewDocumentResource.as_view(name, ...)	Converts the class into an actual view function that can be used with the routing system.
NewDocumentResource.create_search_query(...)	
NewDocumentResource.dispatch_request(*args, ...)	Subclasses have to override this method to implement the actual view function code.
NewDocumentResource.get_document_content(...)	

continues on next page

Table 24 – continued from previous page

NewDocumentResource.get_document_data(...)	
NewDocumentResource.get_request_file()	
NewDocumentResource.get_request_query_fields(...)	
NewDocumentResource.post()	
NewDocumentResource.validate_payload(func)	Perform a payload validation on expected model if necessary

app.blueprints.documents.NewDocumentResource.__init__

NewDocumentResource.**__init__**(*api=None*, **args*, ***kwargs*)
Initialize self. See help(type(self)) for accurate signature.

app.blueprints.documents.NewDocumentResource.as_view

classmethod NewDocumentResource.**as_view**(*name*, **class_args*, ***class_kwargs*)
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.create_search_query

NewDocumentResource.**create_search_query**(*query: peewee.ModelSelect*, *request_data: dict*) → peewee.ModelSelect

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.get_document_content

static NewDocumentResource.**get_document_content**(*document_id: int*)

app.blueprints.documents.NewDocumentResource.get_document_data

NewDocumentResource.**get_document_data**(*document_id: int*) → tuple

app.blueprints.documents.NewDocumentResource.get_request_file

NewDocumentResource.**get_request_file**() → dict

app.blueprints.documents.NewDocumentResource.get_request_query_fields

NewDocumentResource.**get_request_query_fields**(*request_data: dict*) → tuple

app.blueprints.documents.NewDocumentResource.post

NewDocumentResource.**post**()

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

SearchDocumentResource.decorators
SearchDocumentResource.document_serializer
SearchDocumentResource.method_decorators
SearchDocumentResource.methods
SearchDocumentResource.provide_automatic_options
SearchDocumentResource.representations
SearchDocumentResource.request_field_name

app.blueprints.documents.SearchDocumentResource.decorators

SearchDocumentResource.**decorators = ()**

app.blueprints.documents.SearchDocumentResource.document_serializer

SearchDocumentResource.**document_serializer = <DocumentSchema(many=False)>**

app.blueprints.documents.SearchDocumentResource.method_decorators

SearchDocumentResource.**method_decorators = []**

app.blueprints.documents.SearchDocumentResource.methods

SearchDocumentResource.**methods = {'POST'}**

app.blueprints.documents.SearchDocumentResource.provide_automatic_options

SearchDocumentResource.**provide_automatic_options = None**

app.blueprints.documents.SearchDocumentResource.representations

SearchDocumentResource.**representations = None**

app.blueprints.documents.SearchDocumentResource.request_field_name

SearchDocumentResource.**request_field_name = 'document'**

Methods

SearchDocumentResource.__init__([api])	Initialize self.
SearchDocumentResource.as_view(name, ...)	Converts the class into an actual view function that can be used with the routing system.
SearchDocumentResource.create_search_query(...)	
SearchDocumentResource.dispatch_request(...)	Subclasses have to override this method to implement the actual view function code.
SearchDocumentResource.get_document_content(...)	
SearchDocumentResource.get_document_data(...)	
SearchDocumentResource.get_request_file()	

continues on next page

Table 26 – continued from previous page

SearchDocumentResource.get_request_query_fields(...)	
SearchDocumentResource.post()	
SearchDocumentResource.validate_payload(func)	Perform a payload validation on expected model if necessary

app.blueprints.documents.SearchDocumentResource.__init__

SearchDocumentResource.**__init__**(*api=None*, **args*, ***kwargs*)
Initialize self. See help(type(self)) for accurate signature.

app.blueprints.documents.SearchDocumentResource.as_view

classmethod SearchDocumentResource.**as_view**(*name*, **class_args*, ***class_kwargs*)
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.create_search_query

SearchDocumentResource.**create_search_query**(*query: peewee.ModelSelect*, *request_data: dict*) → peewee.ModelSelect

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.get_document_content

static SearchDocumentResource.**get_document_content**(*document_id: int*)

app.blueprints.documents.SearchDocumentResource.get_document_data

SearchDocumentResource.**get_document_data**(*document_id: int*) → tuple

app.blueprints.documents.SearchDocumentResource.get_request_file

SearchDocumentResource.**get_request_file**() → dict

app.blueprints.documents.SearchDocumentResource.get_request_query_fields

SearchDocumentResource.**get_request_query_fields**(*request_data: dict*) → tuple

app.blueprints.documents.SearchDocumentResource.post

SearchDocumentResource.**post**()

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*, **class_args*, ***class_kwargs*)
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.

create_search_query(*query: peewee.ModelSelect*, *request_data: dict*) → peewee.ModelSelect

db_model
alias of *app.models.document.Document*

decorators = ()

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.

document_serializer = <DocumentSchema(many=False)>

static get_document_content(*document_id: int*)

get_document_data(*document_id: int*) → tuple

get_request_file() → dict

get_request_query_fields(*request_data: dict*) → tuple

method_decorators = []

methods = None

provide_automatic_options = None

representations = None

request_field_name = 'document'

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*, **class_args*, ***class_kwargs*)
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.

create_search_query(*query: peewee.ModelSelect*, *request_data: dict*) → peewee.ModelSelect

db_model
alias of *app.models.document.Document*

decorators = ()

delete(*document_id: int*)

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.

document_serializer = <DocumentSchema(many=False)>

get(*document_id: int*) → tuple

static get_document_content(*document_id: int*)

get_document_data(*document_id: int*) → tuple

get_request_file() → dict

get_request_query_fields(*request_data: dict*) → tuple

method_decorators = []

methods = {'DELETE', 'GET', 'PUT'}

provide_automatic_options = None

put(*document_id: int*) → tuple

representations = None

request_field_name = 'document'

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*, **class_args*, ***class_kwargs*)
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.

create_search_query(*query: peewee.ModelSelect*, *request_data: dict*) → peewee.ModelSelect

db_model
alias of *app.models.document.Document*

decorators = ()

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.

document_serializer = <DocumentSchema(many=False)>

static get_document_content(*document_id: int*)

get_document_data(*document_id: int*) → tuple

get_request_file() → dict

get_request_query_fields(*request_data: dict*) → tuple

method_decorators = []

methods = {'POST'}

parser = <flask_restx.reqparse.RequestParser object>

post()

provide_automatic_options = None

representations = None

request_field_name = 'document'

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*, **class_args*, ***class_kwargs*)
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.

create_search_query(*query: peewee.ModelSelect*, *request_data: dict*) → peewee.ModelSelect

db_model
alias of *app.models.document.Document*

decorators = ()

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.

document_serializer = <DocumentSchema(many=False)>

static get_document_content(*document_id: int*)

get_document_data(*document_id: int*) → tuple

get_request_file() → dict

get_request_query_fields(*request_data: dict*) → tuple

method_decorators = []

methods = {'POST'}

post()

provide_automatic_options = None

representations = None

request_field_name = 'document'

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
NewRoleResource. *method_decorators*
NewRoleResource.methods
NewRoleResource. *provide_automatic_options*
NewRoleResource.representations
NewRoleResource.role_serializer

app.blueprints.roles.NewRoleResource.decorators

NewRoleResource.**decorators = ()**

app.blueprints.roles.NewRoleResource.method_decorators

NewRoleResource.**method_decorators = []**

app.blueprints.roles.NewRoleResource.methods

NewRoleResource.**methods = {'POST'}**

app.blueprints.roles.NewRoleResource.provide_automatic_options

NewRoleResource.**provide_automatic_options = None**

app.blueprints.roles.NewRoleResource.representations

NewRoleResource.**representations = None**

app.blueprints.roles.NewRoleResource.role_serializer

NewRoleResource.**role_serializer = <RoleSchema(many=False)>**

Methods

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

app.blueprints.roles.NewRoleResource.__init__

NewRoleResource.**__init__**(*api=None*, **args*, ***kwargs*)
Initialize self. See help(type(self)) for accurate signature.

app.blueprints.roles.NewRoleResource.as_view

classmethod NewRoleResource.**as_view**(*name*, **class_args*, ***class_kwargs*)
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.create_search_query

NewRoleResource.**create_search_query**(*query: peewee.ModelSelect*, *request_data: dict*) → peewee.ModelSelect

app.blueprints.roles.NewRoleResource.deserialize_request_data

NewRoleResource.**deserialize_request_data**(***kwargs: dict*) → dict

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.get_request_query_fields

NewRoleResource.**get_request_query_fields**(*request_data: dict*) → tuple

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

RoleBaseResource.decorators
RoleBaseResource.method_decorators
RoleBaseResource.methods
RoleBaseResource.provide_automatic_options
RoleBaseResource.representations
RoleBaseResource.role_serializer

app.blueprints.roles.RoleBaseResource.decorators

RoleBaseResource.**decorators = ()**

app.blueprints.roles.RoleBaseResource.method_decorators

RoleBaseResource.**method_decorators = []**

app.blueprints.roles.RoleBaseResource.methods

RoleBaseResource.**methods = None**

app.blueprints.roles.RoleBaseResource.provide_automatic_options

RoleBaseResource.**provide_automatic_options = None**

app.blueprints.roles.RoleBaseResource.representations

RoleBaseResource.**representations = None**

app.blueprints.roles.RoleBaseResource.role_serializer

RoleBaseResource.**role_serializer = <RoleSchema(many=False)>**

Methods

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

app.blueprints.roles.RoleBaseResource.__init__

RoleBaseResource.**__init__**(*api=None*, **args*, ***kwargs*)
Initialize self. See help(type(self)) for accurate signature.

app.blueprints.roles.RoleBaseResource.as_view

classmethod RoleBaseResource.**as_view**(*name*, **class_args*, ***class_kwargs*)
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.create_search_query

RoleBaseResource.**create_search_query**(*query: peewee.ModelSelect*, *request_data: dict*) → peewee.ModelSelect

app.blueprints.roles.RoleBaseResource.deserialize_request_data

RoleBaseResource.**deserialize_request_data**(***kwargs: dict*) → dict

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.get_request_query_fields

RoleBaseResource.**get_request_query_fields**(*request_data: dict*) → tuple

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

RoleResource.decorators
RoleResource.method_decorators
RoleResource.methods
RoleResource.provide_automatic_options
RoleResource.representations
RoleResource.role_serializer

app.blueprints.roles.RoleResource.decorators

RoleResource.**decorators = ()**

app.blueprints.roles.RoleResource.method_decorators

RoleResource.**method_decorators = []**

app.blueprints.roles.RoleResource.methods

RoleResource.**methods = {'DELETE', 'GET', 'PUT'}**

app.blueprints.roles.RoleResource.provide_automatic_options

RoleResource.**provide_automatic_options = None**

app.blueprints.roles.RoleResource.representations

RoleResource.**representations = None**

app.blueprints.roles.RoleResource.role_serializer

RoleResource.**role_serializer = <RoleSchema(many=False)>**

Methods

RoleResource.__init__([api])	Initialize self.
RoleResource.as_view(name, *class_args, ...)	Converts the class into an actual view function that can be used with the routing system.
RoleResource.create_search_query(query, ...)	
RoleResource.delete(role_id)	
RoleResource.deserialize_request_data(**kwargs)	
RoleResource.dispatch_request(*args, **kwargs)	Subclasses have to override this method to implement the actual view function code.
RoleResource.get(role_id)	
RoleResource.get_request_query_fields(...)	
RoleResource.put(role_id)	
RoleResource.validate_payload(func)	Perform a payload validation on expected model if necessary

app.blueprints.roles.RoleResource.__init__

RoleResource.**__init__**(*api=None*, **args*, ***kwargs*)
Initialize self. See help(type(self)) for accurate signature.

app.blueprints.roles.RoleResource.as_view

classmethod RoleResource.**as_view**(*name*, **class_args*, ***class_kwargs*)
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.create_search_query

RoleResource.**create_search_query**(*query: peewee.ModelSelect*, *request_data: dict*) → peewee.ModelSelect

app.blueprints.roles.RoleResource.delete

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

app.blueprints.roles.RoleResource.deserialize_request_data

RoleResource.**deserialize_request_data**(***kwargs: dict*) → dict

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.get_request_query_fields

RoleResource.**get_request_query_fields**(*request_data: dict*) → 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

RolesSearchResource.decorators
RolesSearchResource.method_decorators
RolesSearchResource.methods
RolesSearchResource.provide_automatic_options
RolesSearchResource.representations
RolesSearchResource.role_serializer

app.blueprints.roles.RolesSearchResource.decorators

RolesSearchResource.**decorators = ()**

app.blueprints.roles.RolesSearchResource.method_decorators

RolesSearchResource.**method_decorators = []**

app.blueprints.roles.RolesSearchResource.methods

RolesSearchResource.**methods = {'POST'}**

app.blueprints.roles.RolesSearchResource.provide_automatic_options

RolesSearchResource.**provide_automatic_options = None**

app.blueprints.roles.RolesSearchResource.representations

RolesSearchResource.**representations = None**

app.blueprints.roles.RolesSearchResource.role_serializer

RolesSearchResource.**role_serializer = <RoleSchema(many=False)>**

Methods

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

app.blueprints.roles.RolesSearchResource.__init__

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

Initialize self. See help(type(self)) for accurate signature.

app.blueprints.roles.RolesSearchResource.as_view

classmethod RolesSearchResource.**as_view**(*name*, **class_args*, ***class_kwargs*)

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.create_search_query

RolesSearchResource.**create_search_query**(*query: peewee.ModelSelect*, *request_data: dict*) → peewee.ModelSelect

app.blueprints.roles.RolesSearchResource.deserialize_request_data

RolesSearchResource.**deserialize_request_data**(***kwargs: dict*) → dict

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.get_request_query_fields

RolesSearchResource.**get_request_query_fields**(*request_data: dict*) → tuple

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*, **class_args*, ***class_kwargs*)
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.

create_search_query(*query: peewee.ModelSelect*, *request_data: dict*) → peewee.ModelSelect

db_model
alias of *app.models.role.Role*

decorators = ()

deserialize_request_data(***kwargs: dict*) → dict

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_request_query_fields(*request_data: dict*) → tuple

method_decorators = []

methods = {'POST'}

post() → tuple

provide_automatic_options = None

representations = None

role_serializer = <RoleSchema(many=False)>

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*, **class_args*, ***class_kwargs*)
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.

create_search_query(*query: peewee.ModelSelect*, *request_data: dict*) → peewee.ModelSelect

db_model
alias of *app.models.role.Role*

decorators = ()

deserialize_request_data(***kwargs: dict*) → dict

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_request_query_fields(*request_data: dict*) → tuple

method_decorators = []

methods = None

provide_automatic_options = None

representations = None

role_serializer = <RoleSchema(many=False)>

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*, **class_args*, ***class_kwargs*)
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.

create_search_query(*query: peewee.ModelSelect*, *request_data: dict*) → peewee.ModelSelect

db_model
alias of *app.models.role.Role*

decorators = ()

delete(*role_id: int*) → tuple

deserialize_request_data(***kwargs: dict*) → dict

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

get_request_query_fields(*request_data: dict*) → tuple

method_decorators = []

methods = {'DELETE', 'GET', 'PUT'}

provide_automatic_options = None

put(*role_id: int*) → tuple

representations = None

role_serializer = <RoleSchema(many=False)>

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*, **class_args*, ***class_kwargs*)
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.

create_search_query(*query: peewee.ModelSelect*, *request_data: dict*) → peewee.ModelSelect

db_model
alias of *app.models.role.Role*

decorators = ()

deserialize_request_data(***kwargs: dict*) → dict

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_request_query_fields(*request_data: dict*) → tuple

method_decorators = []

methods = {'POST'}

post() → tuple

provide_automatic_options = None

representations = None

role_serializer = <RoleSchema(many=False)>

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

TaskResource.decorators
TaskResource.method_decorators
TaskResource.methods
TaskResource.provide_automatic_options
TaskResource.representations

app.blueprints.tasks.TaskResource.decorators

TaskResource.**decorators = ()**

app.blueprints.tasks.TaskResource.method_decorators

TaskResource.**method_decorators = []**

app.blueprints.tasks.TaskResource.methods

TaskResource.**methods = None**

app.blueprints.tasks.TaskResource.provide_automatic_options

TaskResource.**provide_automatic_options = None**

app.blueprints.tasks.TaskResource.representations

TaskResource.**representations = None**

Methods

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

app.blueprints.tasks.TaskResource.__init__

TaskResource.**__init__**(*api=None*, **args*, ***kwargs*)
Initialize self. See help(type(self)) for accurate signature.

app.blueprints.tasks.TaskResource.as_view

classmethod TaskResource.**as_view**(*name*, **class_args*, ***class_kwargs*)
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.get_task

static TaskResource.**get_task**(*task_id: str*) → celery.local.PromiseProxy

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

TaskStatusResource.decorators
TaskStatusResource.method_decorators
TaskStatusResource.methods
TaskStatusResource.provide_automatic_options
TaskStatusResource.representations

app.blueprints.tasks.TaskStatusResource.decorators

TaskStatusResource.**decorators = ()**

app.blueprints.tasks.TaskStatusResource.method_decorators

TaskStatusResource.**method_decorators = []**

app.blueprints.tasks.TaskStatusResource.methods

TaskStatusResource.**methods = {'GET'}**

app.blueprints.tasks.TaskStatusResource.provide_automatic_options

TaskStatusResource.**provide_automatic_options = None**

app.blueprints.tasks.TaskStatusResource.representations

TaskStatusResource.**representations = None**

Methods

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

app.blueprints.tasks.TaskStatusResource.__init__

TaskStatusResource.**__init__**(*api=None*, **args*, ***kwargs*)
Initialize self. See help(type(self)) for accurate signature.

app.blueprints.tasks.TaskStatusResource.as_view

classmethod TaskStatusResource.**as_view**(*name*, **class_args*, ***class_kwargs*)
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.get_task

static TaskStatusResource.**get_task**(*task_id: str*) → celery.local.PromiseProxy

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*, **class_args*, ***class_kwargs*)
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 = ()

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.

static get_task(*task_id: str*) → celery.local.PromiseProxy

method_decorators = []

methods = None

provide_automatic_options = None

representations = None

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*, **class_args*, ***class_kwargs*)
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 = ()

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*)

static get_task(*task_id: str*) → celery.local.PromiseProxy

method_decorators = []

methods = {'GET'}

provide_automatic_options = None

representations = None

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

ExportUsersExcelAndWordResource.decorators
ExportUsersExcelAndWordResource.method_decorators
ExportUsersExcelAndWordResource.methods
ExportUsersExcelAndWordResource.provide_automatic_options
ExportUsersExcelAndWordResource.representations

continues on next page

Table 42 – continued from previous page

ExportUsersExcelAndWordResource.user_serializer

app.blueprints.users.ExportUsersExcelAndWordResource.decorators

ExportUsersExcelAndWordResource.**decorators = ()**

app.blueprints.users.ExportUsersExcelAndWordResource.method_decorators

ExportUsersExcelAndWordResource.**method_decorators = []**

app.blueprints.users.ExportUsersExcelAndWordResource.methods

ExportUsersExcelAndWordResource.**methods = {'POST'}**

app.blueprints.users.ExportUsersExcelAndWordResource.provide_automatic_options

ExportUsersExcelAndWordResource.**provide_automatic_options = None**

app.blueprints.users.ExportUsersExcelAndWordResource.representations

ExportUsersExcelAndWordResource.**representations = None**

app.blueprints.users.ExportUsersExcelAndWordResource.user_serializer

ExportUsersExcelAndWordResource.**user_serializer = <UserSchema(many=False)>**

Methods

ExportUsersExcelAndWordResource.__init__([api])	Initialize self.
ExportUsersExcelAndWordResource.as_view(...)	Converts the class into an actual view function that can be used with the routing system.
ExportUsersExcelAndWordResource.create_search_query(...)	
ExportUsersExcelAndWordResource.deserialize_request_data(...)	
ExportUsersExcelAndWordResource.dispatch_request(...)	Subclasses have to override this method to implement the actual view function code.
ExportUsersExcelAndWordResource.get_request_query_fields(...)	
ExportUsersExcelAndWordResource.post()	

continues on next page

Table 43 – continued from previous page

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

app.blueprints.users.ExportUsersExcelAndWordResource.__init__

ExportUsersExcelAndWordResource.**__init__**(*api=None*, **args*, ***kwargs*)
Initialize self. See help(type(self)) for accurate signature.

app.blueprints.users.ExportUsersExcelAndWordResource.as_view

classmethod ExportUsersExcelAndWordResource.**as_view**(*name*, **class_args*, ***class_kwargs*)
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.create_search_query

ExportUsersExcelAndWordResource.**create_search_query**(*query: peewee.ModelSelect*, *request_data: dict*) → peewee.ModelSelect

app.blueprints.users.ExportUsersExcelAndWordResource.deserialize_request_data

ExportUsersExcelAndWordResource.**deserialize_request_data**(***kwargs: dict*) → dict

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.get_request_query_fields

ExportUsersExcelAndWordResource.**get_request_query_fields**(*request_data: dict*) → tuple

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

ExportUsersExcelResource.decorators
ExportUsersExcelResource.method_decorators
ExportUsersExcelResource.methods
ExportUsersExcelResource.provide_automatic_options
ExportUsersExcelResource.representations
ExportUsersExcelResource.user_serializer

app.blueprints.users.ExportUsersExcelResource.decorators

ExportUsersExcelResource.**decorators = ()**

app.blueprints.users.ExportUsersExcelResource.method_decorators

ExportUsersExcelResource.**method_decorators = []**

app.blueprints.users.ExportUsersExcelResource.methods

ExportUsersExcelResource.**methods = {'POST'}**

app.blueprints.users.ExportUsersExcelResource.provide_automatic_options

ExportUsersExcelResource.**provide_automatic_options = None**

app.blueprints.users.ExportUsersExcelResource.representations

ExportUsersExcelResource.**representations = None**

app.blueprints.users.ExportUsersExcelResource.user_serializer

ExportUsersExcelResource.**user_serializer = <UserSchema(many=False)>**

Methods

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

app.blueprints.users.ExportUsersExcelResource.__init__

ExportUsersExcelResource.**__init__**(*api=None*, **args*, ***kwargs*)
Initialize self. See help(type(self)) for accurate signature.

app.blueprints.users.ExportUsersExcelResource.as_view

classmethod ExportUsersExcelResource.**as_view**(*name*, **class_args*, ***class_kwargs*)
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.create_search_query

ExportUsersExcelResource.**create_search_query**(*query: peewee.ModelSelect*, *request_data: dict*) → peewee.ModelSelect

app.blueprints.users.ExportUsersExcelResource.deserialize_request_data

ExportUsersExcelResource.**deserialize_request_data**(***kwargs: dict*) → dict

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.get_request_query_fields

ExportUsersExcelResource.**get_request_query_fields**(*request_data: dict*) → tuple

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

ExportUsersWordResource.decorators
ExportUsersWordResource.method_decorators
ExportUsersWordResource.methods
ExportUsersWordResource.provide_automatic_options
ExportUsersWordResource.representations
ExportUsersWordResource.user_serializer

app.blueprints.users.ExportUsersWordResource.decorators

ExportUsersWordResource.**decorators = ()**

app.blueprints.users.ExportUsersWordResource.method_decorators

ExportUsersWordResource.**method_decorators = []**

app.blueprints.users.ExportUsersWordResource.methods

ExportUsersWordResource.**methods = {'POST'}**

app.blueprints.users.ExportUsersWordResource.provide_automatic_options

ExportUsersWordResource.**provide_automatic_options = None**

app.blueprints.users.ExportUsersWordResource.representations

ExportUsersWordResource.**representations = None**

app.blueprints.users.ExportUsersWordResource.user_serializer

ExportUsersWordResource.**user_serializer = <UserSchema(many=False)>**

Methods

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

app.blueprints.users.ExportUsersWordResource.__init__

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

Initialize self. See help(type(self)) for accurate signature.

app.blueprints.users.ExportUsersWordResource.as_view

classmethod ExportUsersWordResource.**as_view**(*name*, **class_args*, ***class_kwargs*)

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.create_search_query

ExportUsersWordResource.**create_search_query**(*query: peewee.ModelSelect*, *request_data: dict*) → peewee.ModelSelect

app.blueprints.users.ExportUsersWordResource.deserialize_request_data

ExportUsersWordResource.**deserialize_request_data**(***kwargs: dict*) → dict

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.get_request_query_fields

ExportUsersWordResource.**get_request_query_fields**(*request_data: dict*) → tuple

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

NewUserResource.decorators
NewUserResource.method_decorators
NewUserResource.methods
NewUserResource.provide_automatic_options
NewUserResource.representations
NewUserResource.user_serializer

app.blueprints.users.NewUserResource.decorators

NewUserResource.**decorators = ()**

app.blueprints.users.NewUserResource.method_decorators

NewUserResource.**method_decorators = []**

app.blueprints.users.NewUserResource.methods

NewUserResource.**methods = {'POST'}**

app.blueprints.users.NewUserResource.provide_automatic_options

NewUserResource.**provide_automatic_options = None**

app.blueprints.users.NewUserResource.representations

NewUserResource.**representations = None**

app.blueprints.users.NewUserResource.user_serializer

NewUserResource.**user_serializer = <UserSchema(many=False)>**

Methods

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

app.blueprints.users.NewUserResource.__init__

NewUserResource.**__init__**(*api=None*, **args*, ***kwargs*)
Initialize self. See help(type(self)) for accurate signature.

app.blueprints.users.NewUserResource.as_view

classmethod NewUserResource.**as_view**(*name*, **class_args*, ***class_kwargs*)
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.create_search_query

NewUserResource.**create_search_query**(*query: peewee.ModelSelect*, *request_data: dict*) → peewee.ModelSelect

app.blueprints.users.NewUserResource.deserialize_request_data

NewUserResource.**deserialize_request_data**(***kwargs: dict*) → dict

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.get_request_query_fields

NewUserResource.**get_request_query_fields**(*request_data: dict*) → tuple

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

UserBaseResource.decorators
UserBaseResource.method_decorators
UserBaseResource.methods
UserBaseResource.provide_automatic_options
UserBaseResource.representations
UserBaseResource.user_serializer

app.blueprints.users.UserBaseResource.decorators

UserBaseResource.**decorators = ()**

app.blueprints.users.UserBaseResource.method_decorators

UserBaseResource.**method_decorators = []**

app.blueprints.users.UserBaseResource.methods

UserBaseResource.**methods = None**

app.blueprints.users.UserBaseResource.provide_automatic_options

UserBaseResource.**provide_automatic_options = None**

app.blueprints.users.UserBaseResource.representations

UserBaseResource.**representations = None**

app.blueprints.users.UserBaseResource.user_serializer

UserBaseResource.**user_serializer = <UserSchema(many=False)>**

Methods

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

app.blueprints.users.UserBaseResource.__init__

UserBaseResource.**__init__**(*api=None*, **args*, ***kwargs*)
Initialize self. See help(type(self)) for accurate signature.

app.blueprints.users.UserBaseResource.as_view

classmethod UserBaseResource.**as_view**(*name*, **class_args*, ***class_kwargs*)
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.create_search_query

UserBaseResource.**create_search_query**(*query: peewee.ModelSelect*, *request_data: dict*) → peewee.ModelSelect

app.blueprints.users.UserBaseResource.deserialize_request_data

UserBaseResource.**deserialize_request_data**(***kwargs: dict*) → dict

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.get_request_query_fields

UserBaseResource.**get_request_query_fields**(*request_data: dict*) → tuple

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

UserResource.decorators
UserResource.method_decorators
UserResource.methods
UserResource.provide_automatic_options
UserResource.representations
UserResource.user_serializer

app.blueprints.users.UserResource.decorators

UserResource.**decorators = ()**

app.blueprints.users.UserResource.method_decorators

UserResource.**method_decorators = []**

app.blueprints.users.UserResource.methods

UserResource.**methods = {'DELETE', 'GET', 'PUT'}**

app.blueprints.users.UserResource.provide_automatic_options

UserResource.**provide_automatic_options = None**

app.blueprints.users.UserResource.representations

UserResource.**representations = None**

app.blueprints.users.UserResource.user_serializer

UserResource.**user_serializer = <UserSchema(many=False)>**

Methods

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

app.blueprints.users.UserResource.__init__

UserResource.**__init__**(*api=None*, **args*, ***kwargs*)
Initialize self. See help(type(self)) for accurate signature.

app.blueprints.users.UserResource.as_view

classmethod UserResource.**as_view**(*name*, **class_args*, ***class_kwargs*)
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.create_search_query

UserResource.**create_search_query**(*query: peewee.ModelSelect*, *request_data: dict*) → peewee.ModelSelect

app.blueprints.users.UserResource.delete

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

app.blueprints.users.UserResource.deserialize_request_data

UserResource.**deserialize_request_data**(***kwargs: dict*) → dict

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.get_request_query_fields

UserResource.**get_request_query_fields**(*request_data: dict*) → 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

UsersSearchResource.decorators
UsersSearchResource.method_decorators
UsersSearchResource.methods
UsersSearchResource.provide_automatic_options
UsersSearchResource.representations
UsersSearchResource.user_serializer

app.blueprints.users.UsersSearchResource.decorators

UsersSearchResource.**decorators = ()**

app.blueprints.users.UsersSearchResource.method_decorators

UsersSearchResource.**method_decorators = []**

app.blueprints.users.UsersSearchResource.methods

UsersSearchResource.**methods = {'POST'}**

app.blueprints.users.UsersSearchResource.provide_automatic_options

UsersSearchResource.**provide_automatic_options = None**

app.blueprints.users.UsersSearchResource.representations

UsersSearchResource.**representations = None**

app.blueprints.users.UsersSearchResource.user_serializer

UsersSearchResource.**user_serializer = <UserSchema(many=False)>**

Methods

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

app.blueprints.users.UsersSearchResource.__init__

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

Initialize self. See help(type(self)) for accurate signature.

app.blueprints.users.UsersSearchResource.as_view

classmethod UsersSearchResource.**as_view**(*name*, **class_args*, ***class_kwargs*)

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.create_search_query

UsersSearchResource.**create_search_query**(*query: peewee.ModelSelect*, *request_data: dict*) → peewee.ModelSelect

app.blueprints.users.UsersSearchResource.deserialize_request_data

UsersSearchResource.**deserialize_request_data**(***kwargs: dict*) → dict

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.get_request_query_fields

UsersSearchResource.**get_request_query_fields**(*request_data: dict*) → tuple

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*, **class_args*, ***class_kwargs*)
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.

create_search_query(*query: peewee.ModelSelect*, *request_data: dict*) → peewee.ModelSelect

db_model
alias of *`app.models.user.User`*

decorators = ()

deserialize_request_data(***kwargs: dict*) → dict

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_request_query_fields(*request_data: dict*) → tuple

method_decorators = []

methods = {'POST'}

post() → tuple

provide_automatic_options = None

representations = None

user_serializer = <UserSchema(many=False)>

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*, **class_args*, ***class_kwargs*)
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.

create_search_query(*query: peewee.ModelSelect*, *request_data: dict*) → peewee.ModelSelect

db_model
alias of *app.models.user.User*

decorators = ()

deserialize_request_data(***kwargs: dict*) → dict

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_request_query_fields(*request_data: dict*) → tuple

method_decorators = []

methods = {'POST'}

post() → tuple

provide_automatic_options = None

representations = None

user_serializer = <UserSchema(many=False)>

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*, **class_args*, ***class_kwargs*)
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.

create_search_query(*query: peewee.ModelSelect*, *request_data: dict*) → peewee.ModelSelect

db_model
alias of *app.models.user.User*

decorators = ()

deserialize_request_data(***kwargs: dict*) → dict

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_request_query_fields(*request_data: dict*) → tuple

method_decorators = []

methods = {'POST'}

post() → tuple

provide_automatic_options = None

representations = None

user_serializer = <UserSchema(many=False)>

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*, **class_args*, ***class_kwargs*)
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.

create_search_query(*query: peewee.ModelSelect*, *request_data: dict*) → peewee.ModelSelect

db_model
alias of *app.models.user.User*

decorators = ()

deserialize_request_data(***kwargs: dict*) → dict

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_request_query_fields(*request_data: dict*) → tuple

method_decorators = []

methods = {'POST'}

post() → tuple

provide_automatic_options = None

representations = None

user_serializer = <UserSchema(many=False)>

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*, **class_args*, ***class_kwargs*)
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.

create_search_query(*query: peewee.ModelSelect*, *request_data: dict*) → peewee.ModelSelect

db_model
alias of *app.models.user.User*

decorators = ()

deserialize_request_data(***kwargs: dict*) → dict

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_request_query_fields(*request_data: dict*) → tuple

method_decorators = []

methods = None

provide_automatic_options = None

representations = None

user_serializer = <UserSchema(many=False)>

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*, **class_args*, ***class_kwargs*)
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.

create_search_query(*query: peewee.ModelSelect*, *request_data: dict*) → peewee.ModelSelect

db_model
alias of *app.models.user.User*

decorators = ()

delete(*user_id: int*) → tuple

deserialize_request_data(***kwargs: dict*) → dict

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

get_request_query_fields(*request_data: dict*) → tuple

method_decorators = []

methods = {'DELETE', 'GET', 'PUT'}

provide_automatic_options = None

put(*user_id: int*) → tuple

representations = None

user_serializer = <UserSchema(many=False)>

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*, **class_args*, ***class_kwargs*)
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.

create_search_query(*query: peewee.ModelSelect*, *request_data: dict*) → peewee.ModelSelect

db_model
alias of *app.models.user.User*

decorators = ()

deserialize_request_data(***kwargs: dict*) → dict

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_request_query_fields(*request_data: dict*) → tuple

method_decorators = []

methods = {'POST'}

post() → tuple

provide_automatic_options = None

representations = None

user_serializer = <UserSchema(many=False)>

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

2.1.2 app.celery

Description

Registers Celery tasks.

Modules

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

Export User Data in Excel task

Parameters

- **self** –
- **created_by** –
- **request_data** –

Returns

app.celery.tasks

Description

(task) app.celery.tasks.**create_user_email**(*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**(*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**(*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_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.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**(*created_by: int*, *request_data: dict*, *to_pdf: int*)
Export User Data in Word

Parameters

- **self** –
- **created_by** –
- **request_data** –
- **to_pdf** –

Returns

Classes

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

app.celery.ContextTask

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

Attributes

ContextTask.Request	
ContextTask.Strategy	
ContextTask.abstract	
ContextTask.acks_late	
ContextTask.acks_on_failure_or_timeout	
ContextTask.app	
ContextTask.autoregister	
ContextTask.backend	
ContextTask.default_retry_delay	
ContextTask.expires	
ContextTask.from_config	
ContextTask.ignore_result	
ContextTask.max_retries	
ContextTask.name	
ContextTask.priority	
ContextTask.rate_limit	
ContextTask.reject_on_worker_lost	
ContextTask.request	Get current request object.
ContextTask.request_stack	
ContextTask.resultrepr_maxsize	
ContextTask.send_events	
ContextTask.serializer	
ContextTask.soft_time_limit	
ContextTask.store_errors_even_if_ignored	
ContextTask.throws	
ContextTask.time_limit	
ContextTask.track_started	
ContextTask.trail	
ContextTask.typing	

app.celery.ContextTask.Request

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

app.celery.ContextTask.Strategy

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

app.celery.ContextTask.abstract

```
ContextTask.abstract = True
```

app.celery.ContextTask.acks_late

```
ContextTask.acks_late = False
```

app.celery.ContextTask.acks_on_failure_or_timeout

```
ContextTask.acks_on_failure_or_timeout = True
```

app.celery.ContextTask.app

```
ContextTask.app = <MyCelery __main__>
```

app.celery.ContextTask.autoregister

```
ContextTask.autoregister = True
```

app.celery.ContextTask.backend

```
property ContextTask.backend
```

app.celery.ContextTask.default_retry_delay

```
ContextTask.default_retry_delay = 180
```

app.celery.ContextTask.expires

ContextTask.**expires = None**

app.celery.ContextTask.from_config

ContextTask.**from_config = (('serializer', 'task_serializer'), ('rate_limit', 'task_**

app.celery.ContextTask.ignore_result

ContextTask.**ignore_result = False**

app.celery.ContextTask.max_retries

ContextTask.**max_retries = 3**

app.celery.ContextTask.name

ContextTask.**name = None**

app.celery.ContextTask.priority

ContextTask.**priority = None**

app.celery.ContextTask.rate_limit

ContextTask.**rate_limit = None**

app.celery.ContextTask.reject_on_worker_lost

ContextTask.**reject_on_worker_lost = None**

app.celery.ContextTask.request

property ContextTask.**request**
Get current request object.

app.celery.ContextTask.request_stack

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

app.celery.ContextTask.resultrepr_maxsize

ContextTask.**resultrepr_maxsize = 1024**

app.celery.ContextTask.send_events

ContextTask.**send_events = True**

app.celery.ContextTask.serializer

ContextTask.**serializer = 'json'**

app.celery.ContextTask.soft_time_limit

ContextTask.**soft_time_limit = None**

app.celery.ContextTask.store_errors_even_if_ignored

ContextTask.**store_errors_even_if_ignored = False**

app.celery.ContextTask.throws

ContextTask.**throws = ()**

app.celery.ContextTask.time_limit

ContextTask.**time_limit = None**

app.celery.ContextTask.track_started

ContextTask.**track_started = False**

app.celery.ContextTask.trail

```
ContextTask.trail = True
```

app.celery.ContextTask.typing

```
ContextTask.typing = True
```

Methods

ContextTask.AsyncResult(task_id, **kwargs)	Get AsyncResult instance for the specified task.
ContextTask.__init__()	Initialize self.
ContextTask.add_around(attr, around)	
ContextTask.add_to_chord(sig[, lazy])	Add signature to the chord the current task is a member of.
ContextTask.add_trail(result)	
ContextTask.after_return(status, retval, ...)	Handler called after the task returns.
ContextTask.annotate()	
ContextTask.apply([args, kwargs, link, ...])	Execute this task locally, by blocking until the task returns.
ContextTask.apply_async([args, kwargs, ...])	Apply tasks asynchronously by sending a message.
ContextTask.bind(app)	
ContextTask.chunks(it, n)	Create a `chunks` task for this task.
ContextTask.delay(*args, **kwargs)	Star argument version of *apply_async()*.
ContextTask.map(it)	Create a `xmap` task from `it`.
ContextTask.on_bound(app)	Called when the task is bound to an app.
ContextTask.on_failure(exc, task_id, args, ...)	**param exc** The exception raised by the task.
ContextTask.on_retry(exc, task_id, args, ...)	Retry handler.
ContextTask.on_success(retval, task_id, ...)	Success handler.
ContextTask.pop_request()	
ContextTask.push_request(*args, **kwargs)	
ContextTask.replace(sig)	Replace this task, with a new task inheriting the task id.
ContextTask.retry([args, kwargs, exc, ...])	Retry the task, adding it to the back of the queue.
ContextTask.run(*args, **kwargs)	The body of the task executed by workers.
ContextTask.s(*args, **kwargs)	Create signature.
ContextTask.send_event(type_[, retry, ...])	Send monitoring event message.

continues on next page

Table 61 – continued from previous page

ContextTask.shadow_name(args, kwargs, options)	Override for custom task name in worker logs/monitoring.
ContextTask.si(*args, **kwargs)	Create immutable signature.
ContextTask.signature([args])	Create signature.
ContextTask.signature_from_request([...])	
ContextTask.starmap(it)	Create a `xstarmap` task from `it`.
ContextTask.start_strategy(app, consumer, ...)	
ContextTask.subtask([args])	Create signature.
ContextTask.subtask_from_request([request, ...])	
ContextTask.update_state([task_id, state, meta])	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__`**()

Initialize self. See help(type(self)) for accurate signature.

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.
- **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
- **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.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

Parameters
- **exc** – The exception raised by the task.
- **task_id** – Unique id of the failed task.
- **args** – Original arguments for the task that failed.
- **kwargs** – Original keyword arguments for the task that failed.
- **einfo** – ExceptionInfo instance, containing the traceback

Returns 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

MyCelery.AsyncResult	Create new result instance.
MyCelery.Beat	**celery beat** scheduler application.
MyCelery.GroupResult	Create new group result instance.
MyCelery.IS_WINDOWS	
MyCelery.IS_macOS	
MyCelery.ResultSet	
MyCelery.SYSTEM	
MyCelery.Task	Base task class for this app.
MyCelery.WorkController	Embeddable worker.
MyCelery.Worker	Worker application.
MyCelery.amqp	@amqp.
MyCelery.amqp_cls	
MyCelery.annotations	
MyCelery.backend	Current backend instance.
MyCelery.backend_cls	
MyCelery.builtin_fixups	
MyCelery.conf	Current configuration.
MyCelery.control	@control.
MyCelery.control_cls	
MyCelery.current_task	Instance of task being executed, or None.
MyCelery.current_worker_task	The task currently being executed by a worker or None.
MyCelery.events	@events.
MyCelery.events_cls	

continues on next page

Table 62 – continued from previous page

MyCelery.loader	Current loader instance.
MyCelery.loader_cls	
MyCelery.log	@log.
MyCelery.log_cls	
MyCelery.main	
MyCelery.oid	Universally unique identifier for this app.
MyCelery.on_after_configure	
MyCelery.on_after_finalize	
MyCelery.on_after_fork	
MyCelery.on_configure	
MyCelery.pool	@pool.
MyCelery.producer_pool	
MyCelery.registry_cls	
MyCelery.steps	
MyCelery.task_cls	
MyCelery.tasks	Task registry.
MyCelery.timezone	Current timezone for this app.
MyCelery.user_options	

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

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**

app.celery.MyCelery.oid

MyCelery.**oid**
Universally unique identifier for this app.

app.celery.MyCelery.on_after_configure

MyCelery.**on_after_configure = None**

app.celery.MyCelery.on_after_finalize

MyCelery.**on_after_finalize = None**

app.celery.MyCelery.on_after_fork

MyCelery.**on_after_fork = None**

app.celery.MyCelery.on_configure

MyCelery.**on_configure = None**

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**

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.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**

Methods

MyCelery.__init__([main, loader, backend, ...])	Initialize self.
MyCelery.add_defaults(fun)	Add default configuration from dict d.
MyCelery.add_periodic_task(schedule, sig[, ...])	
MyCelery.autodiscover_tasks([packages, ...])	Auto-discover task modules.
MyCelery.broker_connection([hostname, ...])	Establish a connection to the message broker.
MyCelery.bugreport()	Return information useful in bug reports.
MyCelery.close()	Clean up after the application.
MyCelery.config_from_cmdline(argv[, namespace])	
MyCelery.config_from_envvar(variable_name[, ...])	Read configuration from environment variable.
MyCelery.config_from_object(obj[, silent, ...])	Read configuration from object.

continues on next page

Table 63 – continued from previous page

MyCelery.connection([hostname, userid, ...])	Establish a connection to the message broker.
MyCelery.connection_for_read([url])	Establish connection used for consuming.
MyCelery.connection_for_write([url])	Establish connection used for producing.
MyCelery.connection_or_acquire([connection, ...])	Context used to acquire a connection from the pool.
MyCelery.create_task_cls()	Create a base task class bound to this app.
MyCelery.default_connection([connection, pool])	Context used to acquire a connection from the pool.
MyCelery.default_producer([producer])	Context used to acquire a producer from the pool.
MyCelery.either(default_key, *defaults)	Get key from configuration or use default values.
MyCelery.finalize([auto])	Finalize the app.
MyCelery.gen_task_name(name, module)	New task default automatic naming.
MyCelery.now()	Return the current time and date as a datetime.
MyCelery.on_init()	Optional callback called at init.
MyCelery.prepare_config(c)	Prepare configuration before it is merged with the defaults.
MyCelery.producer_or_acquire([producer])	Context used to acquire a producer from the pool.
MyCelery.register_task(task)	Utility for registering a task-based class.
MyCelery.select_queues([queues])	Select subset of queues.
MyCelery.send_task(name[, args, kwargs, ...])	Send task by name.
MyCelery.set_current()	Make this the current app for this thread.
MyCelery.set_default()	Make this the default app for all threads.
MyCelery.setup_security([...])	Setup the message-signing serializer.
MyCelery.signature(*args, **kwargs)	Return a new `Signature` bound to this app.
MyCelery.subclass_with_self(Class[, name, ...])	Subclass an app-compatible class.
MyCelery.task(*args, **opts)	Decorator to create a task class out of any callable.
MyCelery.uses_utc_timezone()	Check if the application uses the UTC timezone.

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*)

Initialize self. See help(type(self)) for accurate signature.

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
    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.

https://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*)
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.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.

Functions

init_celery(app)

app.celery.init_celery

app.celery.**init_celery**(*app: flask.app.Flask*) → celery.app.base.Celery

Exceptions

TaskFailure

app.celery.TaskFailure

exception app.celery.**TaskFailure**

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'

Strategy = 'celery.worker.strategy:default'

_app = <MyCelery __main__>

_backend = None

_default_request = None

_exec_options = None

classmethod _get_app()

_get_exec_options()

_get_request()
Get current request object.

abstract = True

acks_late = False

acks_on_failure_or_timeout = True

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.
- **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
- **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()`.

```
autoregister = True

property backend
```

classmethod bind(*app*)

chunks(*it*, *n*)
Create a `chunks` task for this task.

default_retry_delay = 180

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

from_config = (('serializer', 'task_serializer'), ('rate_limit', 'task_default_rate_li

ignore_result = False

map(*it*)
Create a `xmap` task from `it`.

max_retries = 3

name = None

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*, *einfo*) → None

Parameters

- **exc** – The exception raised by the task.
- **task_id** – Unique id of the failed task.
- **args** – Original arguments for the task that failed.
- **kwargs** – Original keyword arguments for the task that failed.
- **einfo** – ExceptionInfo instance, containing the traceback

Returns 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

push_request(**args*, ***kwargs*)

rate_limit = None

reject_on_worker_lost = None

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>

resultrepr_maxsize = 1024

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

serializer = '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

starmap(*it*)
Create a `xstarmap` task from `it`.

start_strategy(*app*, *consumer*, ***kwargs*)

store_errors_even_if_ignored = False

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 = ()

time_limit = None

track_started = False

trail = True

typing = True

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()

_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.

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 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.

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

now()
Return the current time and date as a datetime.

oid
Universally unique identifier for this app.

on_after_configure = None

on_after_finalize = None

on_after_fork = None

on_configure = None

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*)
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.

steps = None

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.

timezone
Current timezone for this app.

This is a cached property taking the time zone from the **:setting:`timezone`** setting.

user_options = None

uses_utc_timezone()
Check if the application uses the UTC timezone.

exception app.celery.**TaskFailure**

args

with_traceback()
Exception.with_traceback(tb) – set self.__traceback__ to tb and return self.

app.celery.**init_celery**(*app: flask.app.Flask*) → celery.app.base.Celery

2.1.3 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

2.1.4 app.middleware

Description

WSGI middleware for validating requests content type.

Classes

Middleware(app)	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

Middleware.__init__(app)	Initialize self.

app.middleware.Middleware.__init__

Middleware.**__init__**(*app: flask.app.Flask*)
Initialize self. See help(type(self)) for accurate signature.

class app.middleware.**Middleware**(*app: flask.app.Flask*)
WSGI middleware for checking if the request has a valid content type.

static _parse_content_type(*request_content_type: any*) → str
Content-Type := type "/" subtype *[";" parameter] https://tools.ietf.org/html/rfc1341

2.1.5 app.models

Description

Registers database models.

Modules

app.models.base
app.models.document
app.models.role
app.models.user

app.models.base

Description

Classes

Base(*args, **kwargs)

app.models.base.Base

class app.models.base.**Base**(**args*, ***kwargs*)
Bases: playhouse.flask_utils.FlaskDB.get_model_class.<locals>.BaseModel

Attributes

Base.dirty_fields
Base.id

app.models.base.Base.dirty_fields

property Base.**dirty_fields**

app.models.base.Base.id

Base.**id = <AutoField: Base.id>**

Methods

Base.__init__(*args, **kwargs)	Initialize self.
Base.add_index(*fields, **kwargs)	
Base.alias([alias])	
Base.bind(database[, bind_refs, bind_backrefs])	
Base.bind_ctx(database[, bind_refs, ...])	
Base.bulk_create(model_list[, batch_size])	
Base.bulk_update(model_list, fields[, ...])	
Base.clone()	
Base.coerce([_coerce])	
Base.copy(method)	
Base.create(**query)	
Base.create_table([safe])	
Base.delete()	
Base.delete_by_id(pk)	
Base.delete_instance([recursive, ...])	
Base.dependencies([search_nullable])	
Base.drop_table([safe, drop_sequences])	
Base.filter(*dq_nodes, **filters)	
Base.get(*query, **filters)	
Base.get_by_id(pk)	
Base.get_fields([exclude, include, sort_order])	
Base.get_id()	
Base.get_or_create(**kwargs)	
Base.get_or_none(*query, **filters)	
Base.index(*fields, **kwargs)	
Base.insert([_Model__data])	
Base.insert_from(query, fields)	
Base.insert_many(rows[, fields])	
Base.is_alias()	
Base.is_dirty()	
Base.noop()	
Base.raw(sql, *params)	
Base.replace([_Model__data])	
Base.replace_many(rows[, fields])	

continues on next page

Table 72 – continued from previous page

Base.save(*args, **kwargs)
Base.select(*fields)
Base.set_by_id(key, value)
Base.table_exists()
Base.truncate_table(**options)
Base.unwrap()
Base.update([_Model__data])
Base.validate_model()

app.models.base.Base.__init__

Base.**__init__**(**args*, ***kwargs*)
Initialize self. See help(type(self)) for accurate signature.

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*)

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: list = None*, *include: list = None*, *sort_order: 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

classmethod Base.**raw**(*sql*, **params*)

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 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*)

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: list = None*, *include: list = None*, *sort_order: 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()

classmethod raw(*sql*, **params*)

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)	Initialize self.
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, ...])	
Document.dependencies([search_nullable])	
Document.drop_table([safe, drop_sequences])	
Document.filter(*dq_nodes, **filters)	
Document.get(*query, **filters)	
Document.get_by_id(pk)	
Document.get_fields([exclude, include, ...])	
Document.get_filepath()	
Document.get_id()	

continues on next page

Table 75 – continued from previous page

Document.get_or_create(**kwargs)
Document.get_or_none(*query, **filters)
Document.index(*fields, **kwargs)
Document.insert([_Model__data])
Document.insert_from(query, fields)
Document.insert_many(rows[, fields])
Document.is_alias()
Document.is_dirty()
Document.noop()
Document.raw(sql, *params)
Document.replace([_Model__data])
Document.replace_many(rows[, fields])
Document.save(*args, **kwargs)
Document.select(*fields)
Document.set_by_id(key, value)
Document.table_exists()
Document.truncate_table(**options)
Document.unwrap()
Document.update([_Model__data])
Document.validate_model()

app.models.document.Document.__init__

Document.**__init__**(**args*, ***kwargs*)
Initialize self. See help(type(self)) for accurate signature.

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*)

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: list = None*, *include: list = None*, *sort_order: 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

classmethod Document.**raw**(*sql*, **params*)

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 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*)

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: list = None*, *include: list = None*, *sort_order: 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()

classmethod raw(*sql*, **params*)

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)	Initialize self.
Role.add_index(*fields, **kwargs)	
Role.add_permissions(permissions)	Add one or more permissions to role.
Role.alias([alias])	
Role.bind(database[, bind_refs, bind_backrefs])	
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)	
Role.get_by_id(pk)	
Role.get_fields([exclude, include, sort_order])	
Role.get_id()	
Role.get_or_create(**kwargs)	
Role.get_or_none(*query, **filters)	
Role.get_permissions()	Return set of permissions associated with role.
Role.index(*fields, **kwargs)	
Role.insert([_Model__data])	
Role.insert_from(query, fields)	
Role.insert_many(rows[, fields])	
Role.is_alias()	
Role.is_dirty()	
Role.noop()	
Role.raw(sql, *params)	
Role.remove_permissions(permissions)	Remove one or more permissions from role.
Role.replace([_Model__data])	
Role.replace_many(rows[, fields])	
Role.save(*args, **kwargs)	
Role.select(*fields)	

continues on next page

Table 78 – continued from previous page

Role.set_by_id(key, value)
Role.table_exists()
Role.truncate_table(**options)
Role.unwrap()
Role.update([_Model__data])
Role.validate_model()

app.models.role.Role.__init__

Role.**__init__**(**args*, ***kwargs*)
Initialize self. See help(type(self)) for accurate signature.

app.models.role.Role.add_index

classmethod Role.**add_index**(**fields*, ***kwargs*)

app.models.role.Role.add_permissions

Role.**add_permissions**(*permissions*)
Add one or more permissions to role.
Parameters **permissions** – a set, list, or single string.
Caller must commit to DB.

New in version 3.3.0.

Deprecated since version 3.4.4: Use `UserDatastore.remove_permissions_from_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*)

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: list = None*, *include: list = None*, *sort_order: 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**()

Return set of permissions associated with role.

Either takes a comma separated string of permissions or an interable of strings if permissions are in their own table.

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

classmethod Role.**raw**(*sql*, **params*)

app.models.role.Role.remove_permissions

Role.**remove_permissions**(*permissions*)

Remove one or more permissions from role.

Parameters **permissions** – a set, list, or single string.

Caller must commit to DB.

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 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*)
Add one or more permissions to role.

Parameters **permissions** – a set, list, or single string.

Caller must commit to DB.

New in version 3.3.0.

Deprecated since version 3.4.4: Use UserDatastore.remove_permissions_from_role()

classmethod alias(*alias=None*)

classmethod bind(*database*, *bind_refs=True*, *bind_backrefs=True*)

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: list = None*, *include: list = None*, *sort_order: list = None*) → set

get_id()

classmethod get_or_create(***kwargs*)

classmethod get_or_none(**query*, ***filters*)

get_permissions()

Return set of permissions associated with role.

Either takes a comma separated string of permissions or an interable of strings if permissions are in their own table.

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()

classmethod raw(*sql*, **params*)

remove_permissions(*permissions*)

Remove one or more permissions from role.

Parameters **permissions** – a set, list, or single string.

Caller must commit to DB.

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

User(*args, **kwargs)

app.models.user.User

class app.models.user.**User**(**args*, ***kwargs*)
Bases: *app.models.base.Base*, flask_security.core.UserMixin

Attributes

User.active	
User.birth_date	
User.children	
User.created_at	
User.created_by	
User.created_by_id	
User.deleted_at	
User.dirty_fields	
User.document_set	
User.email	
User.genre	
User.id	
User.is_active	Returns *True* if the user is active.
User.is_anonymous	
User.is_authenticated	
User.last_name	
User.name	
User.password	
User.roles	
User.updated_at	
User.userrolethrough_set	

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.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**
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)	Initialize self.
User.add_index(*fields, **kwargs)	
User.alias([alias])	
User.bind(database[, bind_refs, bind_backrefs])	
User.bind_ctx(database[, bind_refs, ...])	
User.bulk_create(model_list[, batch_size])	
User.bulk_update(model_list, fields[, ...])	
User.calc_username()	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()	
User.delete_by_id(pk)	
User.delete_instance([recursive, ...])	
User.dependencies([search_nullable])	
User.drop_table([safe, drop_sequences])	
User.ensure_password(plain_text)	
User.filter(*dq_nodes, **filters)	
User.get(*query, **filters)	
User.get_auth_token()	Constructs the user's authentication token.
User.get_by_id(pk)	
User.get_fields([exclude, include, sort_order])	
User.get_id()	Returns the user identification attribute.
User.get_or_create(**kwargs)	
User.get_or_none(*query, **filters)	
User.get_redirect_qparams([existing])	Return user info that will be added to redirect query params.
User.get_reset_token()	
User.get_security_payload()	Serialize user object as response payload.
User.has_permission(permission)	Returns *True* if user has this permission (via a role it has).
User.has_role(role)	Returns *True* if the user identifies with the specified role.
User.index(*fields, **kwargs)	
User.insert([_Model__data])	
User.insert_from(query, fields)	
User.insert_many(rows[, fields])	
User.is_alias()	
User.is_dirty()	
User.noop()	

continues on next page

Table 81 – continued from previous page

User.raw(sql, *params)	
User.replace([_Model__data])	
User.replace_many(rows[, fields])	
User.save(*args, **kwargs)	
User.select(*fields)	
User.set_by_id(key, value)	
User.table_exists()	
User.tf_send_security_token(method, **kwargs)	Generate and send the security code for two-factor.
User.truncate_table(**options)	
User.unwrap()	
User.update([_Model__data])	
User.us_send_security_token(method, **kwargs)	Generate and send the security code for unified sign in.
User.validate_model()	
User.verify_and_update_password(password)	Return `True` if the password is valid for the specified user.
User.verify_auth_token(data)	Perform additional verification of contents of auth token.
User.verify_reset_token(token)	

app.models.user.User.__init__

User.**__init__**(**args*, ***kwargs*)

Initialize self. See help(type(self)) for accurate signature.

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*)

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**()

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**()
Constructs the user's authentication token.

This data MUST be securely signed using the `remember_token_serializer`

app.models.user.User.get_by_id

classmethod User.**get_by_id**(*pk*)

app.models.user.User.get_fields

classmethod User.**get_fields**(*exclude: list = None*, *include: list = None*, *sort_order: list = None*) → set

app.models.user.User.get_id

User.**get_id**()
Returns the user identification attribute.

This will be *fs_uniquifier* if that is available, else base class id (which is via Flask-Login and is user.id).

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=None*)
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.
New in version 3.2.0.

app.models.user.User.get_reset_token

User.**get_reset_token**() → str

app.models.user.User.get_security_payload

User.**get_security_payload**()
Serialize user object as response payload.

app.models.user.User.has_permission

User.**has_permission**(*permission*)
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*)
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

classmethod User.**raw**(*sql*, **params*)

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*, ***kwargs*)

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*, ***kwargs*)

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*)

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*)

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.

app.models.user.User.verify_reset_token

static User.**verify_reset_token**(*token: str*) → any

class app.models.user.**User**(**args*, ***kwargs*)

DoesNotExist

alias of `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*)

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()
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*)

genre = <FixedCharField: User.genre>

classmethod get(**query*, ***filters*)

get_auth_token()
Constructs the user's authentication token.

This data MUST be securely signed using the `remember_token_serializer`

classmethod get_by_id(*pk*)

classmethod get_fields(*exclude: list = None*, *include: list = None*, *sort_order: list = None*) → set

get_id()
Returns the user identification attribute.

This will be *fs_uniquifier* if that is available, else base class id (which is via Flask-Login and is user.id).

New in version 3.4.0.

classmethod get_or_create(***kwargs*)

classmethod get_or_none(**query*, ***filters*)

get_redirect_qparams(*existing=None*)
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.

New in version 3.2.0.

get_reset_token() → str

get_security_payload()
Serialize user object as response payload.

has_permission(*permission*)
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*)
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
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>

classmethod raw(*sql*, **params*)

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*, ***kwargs*)
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*, ***kwargs*)
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*)
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*)
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.

static verify_reset_token(*token: str*) → any

Functions

get_models()

app.models.get_models

app.models.**get_models**() → list

app.models.**get_models**() → list

2.1.6 app.utils

Description

Collection of functions and classes which make common patterns shorter and easier.

Modules

app.utils.decorators
app.utils.file_storage
app.utils.libreoffice
app.utils.marshmallow_schema
app.utils.swagger_models

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__()	Initialize self.
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__**()

Initialize self. See help(type(self)) for accurate signature.

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.marshmallow_schema

Description

Classes

DocumentSchema(*, only, Set[str]] = None, ...)	
ExportWordInputSchema(*, only, ...)	
GetDocumentDataInputSchema(*, only, ...)	
RoleSchema(*, only, Set[str]] = None, ...)	
SearchSchema(*, only, Set[str]] = None, ...)	
Timestamp(*, default, missing, data_key, ...)	Field that serializes to timestamp integer and deserializes to a datetime.datetime class.
UserSchema(*, only, Set[str]] = None, ...)	

app.utils.marshmallow_schema.DocumentSchema

class app.utils.marshmallow_schema.**DocumentSchema**(*, *only: Union[Sequence[str], Set[str]] = None*, *exclude: Union[Sequence[str], Set[str]] = ()*, *many: bool = False*, *context: 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: str = None*)

Bases: flask_marshmallow.schema.Schema

Attributes

DocumentSchema.TYPE_MAPPING
DocumentSchema.dict_class
DocumentSchema.error_messages
DocumentSchema.opts
DocumentSchema.set_class

app.utils.marshmallow_schema.DocumentSchema.TYPE_MAPPING

DocumentSchema.**TYPE_MAPPING = {<class 'str'>: <class 'marshmallow.fields.String'>,**

app.utils.marshmallow_schema.DocumentSchema.dict_class

property DocumentSchema.**dict_class**

app.utils.marshmallow_schema.DocumentSchema.error_messages

DocumentSchema.**error_messages = {}**

app.utils.marshmallow_schema.DocumentSchema.opts

DocumentSchema.**opts = <marshmallow.schema.SchemaOpts object>**

app.utils.marshmallow_schema.DocumentSchema.set_class

property DocumentSchema.**set_class**

Methods

DocumentSchema.__init__(*[, only, exclude, ...])	Initialize self.
DocumentSchema.dump(obj, *[, many])	Serialize an object to native Python data types according to this Schema's fields.
DocumentSchema.dumps(obj, *args[, many])	Same as *dump()*, except return a JSON-encoded string.
DocumentSchema.from_dict(fields, *[, name])	Generate a *Schema* class given a dictionary of fields.
DocumentSchema.get_attribute(obj, attr, default)	Defines how to pull values from an object to serialize.
DocumentSchema.handle_error(error, data, *, ...)	Custom error handler function for the schema.
DocumentSchema.jsonify(obj[, many])	Return a JSON response containing the serialized data.
DocumentSchema.load(data, *[, many, ...])	Deserialize a data structure to an object defined by this Schema's fields.
DocumentSchema.loads(json_data, *[, many, ...])	Same as *load()*, except it takes a JSON string as input.
DocumentSchema.make_url(data, **kwargs)	
DocumentSchema.on_bind_field(field_name, ...)	Hook to modify a field when it is bound to the *Schema*.
DocumentSchema.valid_request_file(data)	
DocumentSchema.validate(data, *[, many, partial])	Validate *data* against the schema, returning a dictionary of validation errors.

app.utils.marshmallow_schema.DocumentSchema.__init__

DocumentSchema.**__init__**(*, *only: Union[Sequence[str], Set[str]] = None*, *exclude: Union[Sequence[str], Set[str]] = ()*, *many: bool = False*, *context: 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: str = None*)

Initialize self. See help(type(self)) for accurate signature.

app.utils.marshmallow_schema.DocumentSchema.dump

DocumentSchema.**dump**(*obj: Any*, *, *many: 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 A dict of serialized data

Return type dict

New in version 1.0.0.

Changed in version 3.0.0b7: This method returns the serialized data rather than a `(data, errors)` duple. A `ValidationError` is raised if `obj` is invalid.

Changed in version 3.0.0rc9: Validation no longer occurs upon serialization.

app.utils.marshmallow_schema.DocumentSchema.dumps

DocumentSchema.**dumps**(*obj: Any*, **args*, *many: 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

Return type str

New in version 1.0.0.

Changed in version 3.0.0b7: This method returns the serialized data rather than a `(data, errors)` duple. A `ValidationError` is raised if `obj` is invalid.

app.utils.marshmallow_schema.DocumentSchema.from_dict

classmethod DocumentSchema.**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.utils.marshmallow_schema.DocumentSchema.get_attribute

DocumentSchema.**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.utils.marshmallow_schema.DocumentSchema.handle_error

DocumentSchema.**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.utils.marshmallow_schema.DocumentSchema.jsonify

DocumentSchema.**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.utils.marshmallow_schema.DocumentSchema.load

DocumentSchema.**load**(*data: Union[Mapping[str, Any], Iterable[Mapping[str, Any]]]*, ***, *many: bool = None*, *partial: Union[bool, Sequence[str], Set[str]] = None*, *unknown: 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)` duple. A `ValidationError` is raised if invalid data are passed.

app.utils.marshmallow_schema.DocumentSchema.loads

`DocumentSchema.`**`loads`**(*json_data: str*, *, *many: bool = None*, *partial: Union[bool, Sequence[str], Set[str]] = None*, *unknown: 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)` duple. A `ValidationError` is raised if invalid data are passed.

app.utils.marshmallow_schema.DocumentSchema.make_url

`DocumentSchema.`**`make_url`**(*data*, ***kwargs*)

app.utils.marshmallow_schema.DocumentSchema.on_bind_field

`DocumentSchema.`**`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.utils.marshmallow_schema.DocumentSchema.valid_request_file

static DocumentSchema.**valid_request_file**(*data*)

app.utils.marshmallow_schema.DocumentSchema.validate

DocumentSchema.**validate**(*data: Mapping*, **, many: bool = None*, *partial: 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.utils.marshmallow_schema.ExportWordInputSchema

class app.utils.marshmallow_schema.**ExportWordInputSchema**(**, only: Union[Sequence[str], Set[str]] = None, exclude: Union[Sequence[str], Set[str]] = (), many: bool = False, context: 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: str = None*)

Bases: flask_marshmallow.schema.Schema

Attributes

ExportWordInputSchema.TYPE_MAPPING
ExportWordInputSchema.dict_class
ExportWordInputSchema.error_messages
ExportWordInputSchema.opts
ExportWordInputSchema.set_class

app.utils.marshmallow_schema.ExportWordInputSchema.TYPE_MAPPING

ExportWordInputSchema.**TYPE_MAPPING = {<class 'str'>: <class 'marshmallow.fields.St**

app.utils.marshmallow_schema.ExportWordInputSchema.dict_class

property ExportWordInputSchema.**dict_class**

app.utils.marshmallow_schema.ExportWordInputSchema.error_messages

ExportWordInputSchema.**error_messages = {}**

app.utils.marshmallow_schema.ExportWordInputSchema.opts

ExportWordInputSchema.**opts = <marshmallow.schema.SchemaOpts object>**

app.utils.marshmallow_schema.ExportWordInputSchema.set_class

property ExportWordInputSchema.**set_class**

Methods

ExportWordInputSchema.__init__(*[, only, ...])	Initialize self.
ExportWordInputSchema.convert_to_integer(...)	
ExportWordInputSchema.dump(obj, *[, many])	Serialize an object to native Python data types according to this Schema's fields.
ExportWordInputSchema.dumps(obj, *args[, many])	Same as *dump()*, except return a JSON-encoded string.
ExportWordInputSchema.from_dict(fields, *[, ...])	Generate a *Schema* class given a dictionary of fields.
ExportWordInputSchema.get_attribute(obj, ...)	Defines how to pull values from an object to serialize.

continues on next page

Table 93 – continued from previous page

ExportWordInputSchema.handle_error(error, ...)	Custom error handler function for the schema.
ExportWordInputSchema.jsonify(obj[, many])	Return a JSON response containing the serialized data.
ExportWordInputSchema.load(data, *[, many, ...])	Deserialize a data structure to an object defined by this Schema's fields.
ExportWordInputSchema.loads(json_data, *[, ...])	Same as *load()*, except it takes a JSON string as input.
ExportWordInputSchema.on_bind_field(...)	Hook to modify a field when it is bound to the *Schema*.
ExportWordInputSchema.validate(data, *[, ...])	Validate *data* against the schema, returning a dictionary of validation errors.

app.utils.marshmallow_schema.ExportWordInputSchema.__init__

ExportWordInputSchema.**__init__**(*, *only: Union[Sequence[str], Set[str]] = None*, *exclude: Union[Sequence[str], Set[str]] = ()*, *many: bool = False*, *context: 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: str = None*)

Initialize self. See help(type(self)) for accurate signature.

app.utils.marshmallow_schema.ExportWordInputSchema.convert_to_integer

ExportWordInputSchema.**convert_to_integer**(*value*, *many*, ***kwargs*)

app.utils.marshmallow_schema.ExportWordInputSchema.dump

ExportWordInputSchema.**dump**(*obj: Any*, *, *many: 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 A dict of serialized data

Return type dict

New in version 1.0.0.

Changed in version 3.0.0b7: This method returns the serialized data rather than a `(data, errors)` duple. A `ValidationError` is raised if `obj` is invalid.

Changed in version 3.0.0rc9: Validation no longer occurs upon serialization.

app.utils.marshmallow_schema.ExportWordInputSchema.dumps

ExportWordInputSchema.**dumps**(*obj: Any*, **args*, *many: 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

Return type str

New in version 1.0.0.

Changed in version 3.0.0b7: This method returns the serialized data rather than a `(data, errors)` duple. A `ValidationError` is raised if `obj` is invalid.

app.utils.marshmallow_schema.ExportWordInputSchema.from_dict

classmethod ExportWordInputSchema.**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.utils.marshmallow_schema.ExportWordInputSchema.get_attribute

ExportWordInputSchema.**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.utils.marshmallow_schema.ExportWordInputSchema.handle_error

ExportWordInputSchema.**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.utils.marshmallow_schema.ExportWordInputSchema.jsonify

ExportWordInputSchema.**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.utils.marshmallow_schema.ExportWordInputSchema.load

ExportWordInputSchema.**load**(*data: Union[Mapping[str, Any], Iterable[Mapping[str, Any]]]*, ***, *many: bool = None*, *partial: Union[bool, Sequence[str], Set[str]] = None*, *unknown: 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)` duple. A `ValidationError` is raised if invalid data are passed.

app.utils.marshmallow_schema.ExportWordInputSchema.loads

ExportWordInputSchema.**loads**(*json_data: str*, ***, *many: bool = None*, *partial: Union[bool, Sequence[str], Set[str]] = None*, *unknown: 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)` duple. A `ValidationError` is raised if invalid data are passed.

app.utils.marshmallow_schema.ExportWordInputSchema.on_bind_field

ExportWordInputSchema.**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.utils.marshmallow_schema.ExportWordInputSchema.validate

ExportWordInputSchema.**validate**(*data: Mapping*, ***, *many: bool = None*, *partial: 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.utils.marshmallow_schema.GetDocumentDataInputSchema

class app.utils.marshmallow_schema.**GetDocumentDataInputSchema**(*, *only: Union[Sequence[str], Set[str]] = None*, *exclude: Union[Sequence[str], Set[str]] = ()*, *many: bool = False*, *context: 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: str = None*)

Bases: flask_marshmallow.schema.Schema

Attributes

GetDocumentDataInputSchema.TYPE_MAPPING
GetDocumentDataInputSchema.dict_class
GetDocumentDataInputSchema.error_messages
GetDocumentDataInputSchema.opts
GetDocumentDataInputSchema.set_class

app.utils.marshmallow_schema.GetDocumentDataInputSchema.TYPE_MAPPING

GetDocumentDataInputSchema.**TYPE_MAPPING = {<class 'str'>: <class 'marshmallow.fiel**

app.utils.marshmallow_schema.GetDocumentDataInputSchema.dict_class

property GetDocumentDataInputSchema.**dict_class**

app.utils.marshmallow_schema.GetDocumentDataInputSchema.error_messages

GetDocumentDataInputSchema.**error_messages = {}**

app.utils.marshmallow_schema.GetDocumentDataInputSchema.opts

GetDocumentDataInputSchema.**opts = <marshmallow.schema.SchemaOpts object>**

app.utils.marshmallow_schema.GetDocumentDataInputSchema.set_class

property GetDocumentDataInputSchema.**set_class**

Methods

GetDocumentDataInputSchema.__init__(*[, ...])	Initialize self.
GetDocumentDataInputSchema.convert_to_integer(...)	
GetDocumentDataInputSchema.dump(obj, *[, many])	Serialize an object to native Python data types according to this Schema's fields.
GetDocumentDataInputSchema.dumps(obj, *args)	Same as *dump()*, except return a JSON-encoded string.
GetDocumentDataInputSchema.from_dict(fields, *)	Generate a *Schema* class given a dictionary of fields.
GetDocumentDataInputSchema.get_attribute(...)	Defines how to pull values from an object to serialize.
GetDocumentDataInputSchema.handle_error(...)	Custom error handler function for the schema.
GetDocumentDataInputSchema.jsonify(obj[, many])	Return a JSON response containing the serialized data.
GetDocumentDataInputSchema.load(data, *[, ...])	Deserialize a data structure to an object defined by this Schema's fields.
GetDocumentDataInputSchema.loads(json_data, *)	Same as *load()*, except it takes a JSON string as input.
GetDocumentDataInputSchema.on_bind_field(...)	Hook to modify a field when it is bound to the *Schema*.
GetDocumentDataInputSchema.validate(data, *)	Validate *data* against the schema, returning a dictionary of validation errors.

app.utils.marshmallow_schema.GetDocumentDataInputSchema.__init__

GetDocumentDataInputSchema.**__init__**(*, *only: Union[Sequence[str], Set[str]] = None*, *exclude: Union[Sequence[str], Set[str]] = ()*, *many: bool = False*, *context: 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: str = None*)

Initialize self. See help(type(self)) for accurate signature.

app.utils.marshmallow_schema.GetDocumentDataInputSchema.convert_to_integer

GetDocumentDataInputSchema.**convert_to_integer**(*value*, *many*, ***kwargs*)

app.utils.marshmallow_schema.GetDocumentDataInputSchema.dump

GetDocumentDataInputSchema.**dump**(*obj: Any*, **, many: 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 A dict of serialized data

Return type dict

New in version 1.0.0.

Changed in version 3.0.0b7: This method returns the serialized data rather than a `(data, errors)` duple. A `ValidationError` is raised if `obj` is invalid.

Changed in version 3.0.0rc9: Validation no longer occurs upon serialization.

app.utils.marshmallow_schema.GetDocumentDataInputSchema.dumps

GetDocumentDataInputSchema.**dumps**(*obj: Any*, **args*, *many: 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

Return type str

New in version 1.0.0.

Changed in version 3.0.0b7: This method returns the serialized data rather than a `(data, errors)` duple. A `ValidationError` is raised if `obj` is invalid.

app.utils.marshmallow_schema.GetDocumentDataInputSchema.from_dict

classmethod GetDocumentDataInputSchema.**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.utils.marshmallow_schema.GetDocumentDataInputSchema.get_attribute

GetDocumentDataInputSchema.**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.utils.marshmallow_schema.GetDocumentDataInputSchema.handle_error

GetDocumentDataInputSchema.**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.utils.marshmallow_schema.GetDocumentDataInputSchema.jsonify

GetDocumentDataInputSchema.**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.utils.marshmallow_schema.GetDocumentDataInputSchema.load

GetDocumentDataInputSchema.**load**(*data: Union[Mapping[str, Any], Iterable[Mapping[str, Any]]]*, ***, *many: bool = None*, *partial: Union[bool, Sequence[str], Set[str]] = None*, *unknown: 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)` duple. A `ValidationError` is raised if invalid data are passed.

app.utils.marshmallow_schema.GetDocumentDataInputSchema.loads

GetDocumentDataInputSchema.**loads**(*json_data: str*, ***, *many: bool = None*, *partial: Union[bool, Sequence[str], Set[str]] = None*, *unknown: 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)` duple. A `ValidationError` is raised if invalid data are passed.

app.utils.marshmallow_schema.GetDocumentDataInputSchema.on_bind_field

`GetDocumentDataInputSchema.`**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.utils.marshmallow_schema.GetDocumentDataInputSchema.validate

`GetDocumentDataInputSchema.`**validate**(*data: Mapping*, ***, *many: bool = None*, *partial: 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.utils.marshmallow_schema.RoleSchema

class app.utils.marshmallow_schema.**RoleSchema**(***, *only: Union[Sequence[str], Set[str]] = None*, *exclude: Union[Sequence[str], Set[str]] = ()*, *many: bool = False*, *context: 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: str = None*)

Bases: `flask_marshmallow.schema.Schema`

Attributes

RoleSchema.TYPE_MAPPING
RoleSchema.dict_class
RoleSchema.error_messages
RoleSchema.opts
RoleSchema.set_class

app.utils.marshmallow_schema.RoleSchema.TYPE_MAPPING

RoleSchema.**TYPE_MAPPING = {<class 'str'>: <class 'marshmallow.fields.String'>, <cl**

app.utils.marshmallow_schema.RoleSchema.dict_class

property RoleSchema.**dict_class**

app.utils.marshmallow_schema.RoleSchema.error_messages

RoleSchema.**error_messages = {}**

app.utils.marshmallow_schema.RoleSchema.opts

RoleSchema.**opts = <marshmallow.schema.SchemaOpts object>**

app.utils.marshmallow_schema.RoleSchema.set_class

property RoleSchema.**set_class**

Methods

RoleSchema.__init__(*[, only, exclude, ...])	Initialize self.
RoleSchema.dump(obj, *[, many])	Serialize an object to native Python data types according to this Schema's fields.
RoleSchema.dumps(obj, *args[, many])	Same as *dump()*, except return a JSON-encoded string.
RoleSchema.from_dict(fields, *[, name])	Generate a *Schema* class given a dictionary of fields.
RoleSchema.get_attribute(obj, attr, default)	Defines how to pull values from an object to serialize.
RoleSchema.handle_error(error, data, *, ...)	Custom error handler function for the schema.
RoleSchema.jsonify(obj[, many])	Return a JSON response containing the serialized data.

continues on next page

Table 97 – continued from previous page

RoleSchema.load(data, *[, many, partial, ...])	Deserialize a data structure to an object defined by this Schema's fields.
RoleSchema.loads(json_data, *[, many, ...])	Same as *load()*, except it takes a JSON string as input.
RoleSchema.on_bind_field(field_name, field_obj)	Hook to modify a field when it is bound to the *Schema*.
RoleSchema.valid_request_role(data)	
RoleSchema.validate(data, *[, many, partial])	Validate *data* against the schema, returning a dictionary of validation errors.

app.utils.marshmallow_schema.RoleSchema.__init__

RoleSchema.**__init__**(*, *only: Union[Sequence[str], Set[str]] = None*, *exclude: Union[Sequence[str], Set[str]] = ()*, *many: bool = False*, *context: 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: str = None*)

Initialize self. See help(type(self)) for accurate signature.

app.utils.marshmallow_schema.RoleSchema.dump

RoleSchema.**dump**(*obj: Any*, *, *many: 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 A dict of serialized data

Return type dict

New in version 1.0.0.

Changed in version 3.0.0b7: This method returns the serialized data rather than a `(data, errors)` duple. A `ValidationError` is raised if `obj` is invalid.

Changed in version 3.0.0rc9: Validation no longer occurs upon serialization.

app.utils.marshmallow_schema.RoleSchema.dumps

RoleSchema.**dumps**(*obj: Any*, **args*, *many: 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

Return type str

New in version 1.0.0.

Changed in version 3.0.0b7: This method returns the serialized data rather than a `(data, errors)` duple. A `ValidationError` is raised if `obj` is invalid.

app.utils.marshmallow_schema.RoleSchema.from_dict

classmethod RoleSchema.**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.utils.marshmallow_schema.RoleSchema.get_attribute

RoleSchema.**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.utils.marshmallow_schema.RoleSchema.handle_error

RoleSchema.**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.utils.marshmallow_schema.RoleSchema.jsonify

RoleSchema.**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.utils.marshmallow_schema.RoleSchema.load

RoleSchema.**load**(*data: Union[Mapping[str, Any], Iterable[Mapping[str, Any]]]*, ***, *many: bool = None*, *partial: Union[bool, Sequence[str], Set[str]] = None*, *unknown: 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)` duple. A `ValidationError` is raised if invalid data are passed.

app.utils.marshmallow_schema.RoleSchema.loads

RoleSchema.**loads**(*json_data: str*, ***, *many: bool = None*, *partial: Union[bool, Sequence[str], Set[str]] = None*, *unknown: 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)` duple. A `ValidationError` is raised if invalid data are passed.

app.utils.marshmallow_schema.RoleSchema.on_bind_field

RoleSchema.**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.utils.marshmallow_schema.RoleSchema.valid_request_role

static RoleSchema.**valid_request_role**(*data: dict*) → dict

app.utils.marshmallow_schema.RoleSchema.validate

RoleSchema.**validate**(*data: Mapping*, ***, *many: bool = None*, *partial: 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.utils.marshmallow_schema.SearchSchema

class app.utils.marshmallow_schema.**SearchSchema**(***, *only: Union[Sequence[str], Set[str]] = None*, *exclude: Union[Sequence[str], Set[str]] = ()*, *many: bool = False*, *context: 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: str = None*)

Bases: `flask_marshmallow.schema.Schema`

Attributes

SearchSchema.TYPE_MAPPING
SearchSchema.dict_class
SearchSchema.error_messages
SearchSchema.opts
SearchSchema.set_class

app.utils.marshmallow_schema.SearchSchema.TYPE_MAPPING

SearchSchema.**TYPE_MAPPING = {<class 'str'>: <class 'marshmallow.fields.String'>, <**

app.utils.marshmallow_schema.SearchSchema.dict_class

property SearchSchema.**dict_class**

app.utils.marshmallow_schema.SearchSchema.error_messages

SearchSchema.**error_messages = {}**

app.utils.marshmallow_schema.SearchSchema.opts

SearchSchema.**opts = <marshmallow.schema.SchemaOpts object>**

app.utils.marshmallow_schema.SearchSchema.set_class

property SearchSchema.**set_class**

Methods

SearchSchema.__init__(*[, only, exclude, ...])	Initialize self.
SearchSchema.dump(obj, *[, many])	Serialize an object to native Python data types according to this Schema's fields.
SearchSchema.dumps(obj, *args[, many])	Same as *dump()*, except return a JSON-encoded string.
SearchSchema.from_dict(fields, *[, name])	Generate a *Schema* class given a dictionary of fields.
SearchSchema.get_attribute(obj, attr, default)	Defines how to pull values from an object to serialize.
SearchSchema.handle_error(error, data, *, ...)	Custom error handler function for the schema.
SearchSchema.jsonify(obj[, many])	Return a JSON response containing the serialized data.

continues on next page

Table 99 – continued from previous page

SearchSchema.load(data, *[, many, partial, ...])	Deserialize a data structure to an object defined by this Schema's fields.
SearchSchema.loads(json_data, *[, many, ...])	Same as *load()*, except it takes a JSON string as input.
SearchSchema.on_bind_field(field_name, field_obj)	Hook to modify a field when it is bound to the *Schema*.
SearchSchema.validate(data, *[, many, partial])	Validate *data* against the schema, returning a dictionary of validation errors.

app.utils.marshmallow_schema.SearchSchema.__init__

SearchSchema.**__init__**(*, *only: Union[Sequence[str], Set[str]] = None*, *exclude: Union[Sequence[str], Set[str]] = ()*, *many: bool = False*, *context: 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: str = None*)

Initialize self. See help(type(self)) for accurate signature.

app.utils.marshmallow_schema.SearchSchema.dump

SearchSchema.**dump**(*obj: Any*, *, *many: 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 A dict of serialized data

Return type dict

New in version 1.0.0.

Changed in version 3.0.0b7: This method returns the serialized data rather than a `(data, errors)` duple. A `ValidationError` is raised if `obj` is invalid.

Changed in version 3.0.0rc9: Validation no longer occurs upon serialization.

app.utils.marshmallow_schema.SearchSchema.dumps

SearchSchema.**dumps**(*obj: Any*, **args*, *many: 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

Return type str

New in version 1.0.0.

Changed in version 3.0.0b7: This method returns the serialized data rather than a `(data, errors)` duple. A `ValidationError` is raised if `obj` is invalid.

app.utils.marshmallow_schema.SearchSchema.from_dict

classmethod SearchSchema.**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.utils.marshmallow_schema.SearchSchema.get_attribute

SearchSchema.**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.utils.marshmallow_schema.SearchSchema.handle_error

SearchSchema.**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.utils.marshmallow_schema.SearchSchema.jsonify

SearchSchema.**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.utils.marshmallow_schema.SearchSchema.load

SearchSchema.**load**(*data: Union[Mapping[str, Any], Iterable[Mapping[str, Any]]]*, ***, *many: bool = None*, *partial: Union[bool, Sequence[str], Set[str]] = None*, *unknown: 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)` duple. A `ValidationError` is raised if invalid data are passed.

app.utils.marshmallow_schema.SearchSchema.loads

SearchSchema.**loads**(*json_data: str*, ***, *many: bool = None*, *partial: Union[bool, Sequence[str], Set[str]] = None*, *unknown: 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)` duple. A `ValidationError` is raised if invalid data are passed.

app.utils.marshmallow_schema.SearchSchema.on_bind_field

`SearchSchema.`**`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.utils.marshmallow_schema.SearchSchema.validate

`SearchSchema.`**`validate`**(*data: Mapping*, ***, *many: bool = None*, *partial: 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.utils.marshmallow_schema.Timestamp

class app.utils.marshmallow_schema.**Timestamp**(**, default: Any = <marshmallow.missing>, missing: Any = <marshmallow.missing>, data_key: str = None, attribute: str = None, validate: Union[Callable[[Any], Any], Iterable[Callable[[Any], Any]]] = None, required: bool = False, allow_none: bool = None, load_only: bool = False, dump_only: bool = False, error_messages: Dict[str, str] = None, **metadata*)

Bases: `marshmallow.fields.Field`

Field that serializes to timestamp integer and deserializes to a datetime.datetime class.

Attributes

Timestamp.context	The context dictionary for the parent `Schema`.
Timestamp.default_error_messages	
Timestamp.name	
Timestamp.parent	
Timestamp.root	Reference to the *Schema* that this field belongs to even if it is buried in a container field (e.g.

app.utils.marshmallow_schema.Timestamp.context

property Timestamp.**context**
The context dictionary for the parent `Schema`.

app.utils.marshmallow_schema.Timestamp.default_error_messages

Timestamp.**default_error_messages = {'null': 'Field may not be null.', 'required':**

app.utils.marshmallow_schema.Timestamp.name

Timestamp.**name = None**

app.utils.marshmallow_schema.Timestamp.parent

Timestamp.**parent = None**

app.utils.marshmallow_schema.Timestamp.root

property Timestamp.**root**
Reference to the *Schema* that this field belongs to even if it is buried in a container field (e.g. *List*). Return *None* for unbound fields.

Methods

Timestamp.__init__(*[, default, missing, . . .])	Initialize self.
Timestamp.deserialize(value[, attr, data])	Deserialize `value`.
Timestamp.fail(key, **kwargs)	Helper method that raises a *ValidationError* with an error message from `self.error_messages`.
Timestamp.get_value(obj, attr[, accessor, . . .])	Return the value for a given key from an object.

continues on next page

Table 101 – continued from previous page

Timestamp.make_error(key, **kwargs)	Helper method to make a *ValidationError* with an error message from `self.error_messages`.
Timestamp.serialize(attr, obj[, accessor])	Pulls the value for the given key from the object, applies the field's formatting and returns the result.

app.utils.marshmallow_schema.Timestamp.__init__

Timestamp.**__init__**(**, default: Any = <marshmallow.missing>, missing: Any = <marshmallow.missing>, data_key: str = None, attribute: str = None, validate: Union[Callable[[Any], Any], Iterable[Callable[[Any], Any]]] = None, required: bool = False, allow_none: bool = None, load_only: bool = False, dump_only: bool = False, error_messages: Dict[str, str] = None, **metadata*) → None

Initialize self. See help(type(self)) for accurate signature.

app.utils.marshmallow_schema.Timestamp.deserialize

Timestamp.**deserialize**(*value: Any*, *attr: str = None*, *data: 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.utils.marshmallow_schema.Timestamp.fail

Timestamp.**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.utils.marshmallow_schema.Timestamp.get_value

Timestamp.**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.utils.marshmallow_schema.Timestamp.make_error

Timestamp.**make_error**(*key: str*, ***kwargs*) → marshmallow.exceptions.ValidationError
Helper method to make a *ValidationError* with an error message from `self.error_messages`.

app.utils.marshmallow_schema.Timestamp.serialize

Timestamp.**serialize**(*attr: str*, *obj: Any*, *accessor: 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.utils.marshmallow_schema.UserSchema

class app.utils.marshmallow_schema.**UserSchema**(**, only: Union[Sequence[str], Set[str]] = None, exclude: Union[Sequence[str], Set[str]] = (), many: bool = False, context: 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: str = None*)

Bases: `flask_marshmallow.schema.Schema`

Attributes

UserSchema.TYPE_MAPPING
UserSchema.dict_class
UserSchema.error_messages
UserSchema.opts
UserSchema.set_class

app.utils.marshmallow_schema.UserSchema.TYPE_MAPPING

UserSchema.**TYPE_MAPPING = {<class 'str'>: <class 'marshmallow.fields.String'>, <cl**

app.utils.marshmallow_schema.UserSchema.dict_class

property UserSchema.**dict_class**

app.utils.marshmallow_schema.UserSchema.error_messages

UserSchema.**error_messages = {}**

app.utils.marshmallow_schema.UserSchema.opts

UserSchema.**opts = <marshmallow.schema.SchemaOpts object>**

app.utils.marshmallow_schema.UserSchema.set_class

property UserSchema.**set_class**

Methods

UserSchema.__init__(*[, only, exclude, ...])	Initialize self.
UserSchema.dump(obj, *[, many])	Serialize an object to native Python data types according to this Schema's fields.
UserSchema.dumps(obj, *args[, many])	Same as *dump()*, except return a JSON-encoded string.
UserSchema.from_dict(fields, *[, name])	Generate a *Schema* class given a dictionary of fields.
UserSchema.get_attribute(obj, attr, default)	Defines how to pull values from an object to serialize.
UserSchema.handle_error(error, data, *, ...)	Custom error handler function for the schema.
UserSchema.jsonify(obj[, many])	Return a JSON response containing the serialized data.
UserSchema.load(data, *[, many, partial, ...])	Deserialize a data structure to an object defined by this Schema's fields.
UserSchema.loads(json_data, *[, many, ...])	Same as *load()*, except it takes a JSON string as input.
UserSchema.on_bind_field(field_name, field_obj)	Hook to modify a field when it is bound to the *Schema*.
UserSchema.valid_request_user(data)	
UserSchema.validate(data, *[, many, partial])	Validate *data* against the schema, returning a dictionary of validation errors.

continues on next page

Table 103 – continued from previous page

UserSchema.validate_credentials(data)
UserSchema.validate_email(data)
UserSchema.validate_password(password)

app.utils.marshmallow_schema.UserSchema.__init__

UserSchema.**__init__**(*, *only: Union[Sequence[str], Set[str]] = None*, *exclude: Union[Sequence[str], Set[str]] = ()*, *many: bool = False*, *context: 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: str = None*)

Initialize self. See help(type(self)) for accurate signature.

app.utils.marshmallow_schema.UserSchema.dump

UserSchema.**dump**(*obj: Any*, *, *many: 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 A dict of serialized data

Return type dict

New in version 1.0.0.

Changed in version 3.0.0b7: This method returns the serialized data rather than a `(data, errors)` duple. A `ValidationError` is raised if `obj` is invalid.

Changed in version 3.0.0rc9: Validation no longer occurs upon serialization.

app.utils.marshmallow_schema.UserSchema.dumps

UserSchema.**dumps**(*obj: Any*, **args*, *many: 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

Return type str

New in version 1.0.0.

Changed in version 3.0.0b7: This method returns the serialized data rather than a `(data, errors)` duple. A `ValidationError` is raised if `obj` is invalid.

app.utils.marshmallow_schema.UserSchema.from_dict

classmethod UserSchema.**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.utils.marshmallow_schema.UserSchema.get_attribute

UserSchema.**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.utils.marshmallow_schema.UserSchema.handle_error

UserSchema.**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.utils.marshmallow_schema.UserSchema.jsonify

UserSchema.**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.utils.marshmallow_schema.UserSchema.load

UserSchema.**load**(*data: Union[Mapping[str, Any], Iterable[Mapping[str, Any]]]*, ***, *many: bool = None*, *partial: Union[bool, Sequence[str], Set[str]] = None*, *unknown: 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)` duple. A `ValidationError` is raised if invalid data are passed.

app.utils.marshmallow_schema.UserSchema.loads

UserSchema.**loads**(*json_data: str*, ***, *many: bool = None*, *partial: Union[bool, Sequence[str], Set[str]] = None*, *unknown: 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)` duple. A `ValidationError` is raised if invalid data are passed.

app.utils.marshmallow_schema.UserSchema.on_bind_field

`UserSchema.`**`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.utils.marshmallow_schema.UserSchema.valid_request_user

`static` `UserSchema.`**`valid_request_user`**(*data: dict*) → dict

app.utils.marshmallow_schema.UserSchema.validate

`UserSchema.`**`validate`**(*data: Mapping*, ***, *many: bool = None*, *partial: 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.utils.marshmallow_schema.UserSchema.validate_credentials

`UserSchema.`**`validate_credentials`**(*data: dict*) → dict

app.utils.marshmallow_schema.UserSchema.validate_email

`static` `UserSchema.`**`validate_email`**(*data: dict*) → dict

app.utils.marshmallow_schema.UserSchema.validate_password

static UserSchema.**validate_password**(*password: str*) → None

class app.utils.marshmallow_schema.**DocumentSchema**(*, *only: Union[Sequence[str], Set[str]] = None*, *exclude: Union[Sequence[str], Set[str]] = ()*, *many: bool = False*, *context: 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: str = None*)

class Meta

fields = ('id', 'name', 'mime_type', 'size', 'url', 'created_at', 'updated_at', 'de

ordered = True

OPTIONS_CLASS
alias of marshmallow.schema.SchemaOpts

TYPE_MAPPING = {<class 'str'>: <class 'marshmallow.fields.String'>, <class 'bytes'>:

_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.Timestamp(default=<marshmallow.missing>, at

_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[_T, List[_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: bool = None*, *partial: Union[bool, Sequence[str], Set[str]] = None*, *unknown: 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 = {('post_dump', False): ['make_url']}

_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[_T, Iterable[_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

dump(*obj: Any*, *, *many: 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 A dict of serialized data

Return type dict

New in version 1.0.0.

Changed in version 3.0.0b7: This method returns the serialized data rather than a `(data, errors)` duple. 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: 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

Return type str

New in version 1.0.0.

Changed in version 3.0.0b7: This method returns the serialized data rather than a `(data, errors)` duple. A `ValidationError` is raised if `obj` is invalid.

error_messages = {}

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: bool = None*, *partial: Union[bool, Sequence[str], Set[str]] = None*, *unknown: 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)` duple. A `ValidationError` is raised if invalid data are passed.

loads(*json_data: str*, ***, *many: bool = None*, *partial: Union[bool, Sequence[str], Set[str]] = None*, *unknown: 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)` duple. A `ValidationError` is raised if invalid data are passed.

make_url(*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

static valid_request_file(*data*)

validate(*data: Mapping*, ***, *many: bool = None*, *partial: 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.utils.marshmallow_schema.**ExportWordInputSchema**(*, *only: Union[Sequence[str], Set[str]] = None*, *exclude: Union[Sequence[str], Set[str]] = ()*, *many: bool = False*, *context: 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: 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.
- **`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'>:

_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(default=<marshmallow.missing>, attribut

_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[_T, List[_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: bool = None*, *partial: Union[bool, Sequence[str], Set[str]] = None*, *unknown: 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): ['convert_to_integer']}

_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[_T, Iterable[_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`.

convert_to_integer(*value*, *many*, ***kwargs*)

property dict_class

dump(*obj: Any*, ***, *many: 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 A dict of serialized data

Return type dict

New in version 1.0.0.

Changed in version 3.0.0b7: This method returns the serialized data rather than a `(data, errors)` duple. 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: 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

Return type str

New in version 1.0.0.

Changed in version 3.0.0b7: This method returns the serialized data rather than a `(data, errors)` duple. A `ValidationError` is raised if `obj` is invalid.

error_messages = {}

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: bool = None*, *partial: Union[bool, Sequence[str], Set[str]] = None*, *unknown: 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)` duple. A `ValidationError` is raised if invalid data are passed.

loads(*json_data: str*, ***, *many: bool = None*, *partial: Union[bool, Sequence[str], Set[str]] = None*, *unknown: 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)` duple. 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

validate(*data: Mapping*, **, many: bool = None*, *partial: 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.utils.marshmallow_schema.**GetDocumentDataInputSchema**(**, only: Union[Sequence[str], Set[str]] = None, exclude: Union[Sequence[str], Set[str]] = (), many: bool = False, context: 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: 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.
- **`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'>:`

`_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(default=<marshmallow.missing>, a

_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[_T, List[_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: bool = None*, *partial: Union[bool, Sequence[str], Set[str]] = None*, *unknown: 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): ['convert_to_integer']}

_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[_T, Iterable[_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`.

convert_to_integer(*value*, *many*, ***kwargs*)

property dict_class

dump(*obj: Any*, *, *many: 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 A dict of serialized data

Return type dict

New in version 1.0.0.

Changed in version 3.0.0b7: This method returns the serialized data rather than a `(data, errors)` duple. 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: 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

Return type str

New in version 1.0.0.

Changed in version 3.0.0b7: This method returns the serialized data rather than a `(data, errors)` duple. A `ValidationError` is raised if `obj` is invalid.

error_messages = {}

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: bool = None*, *partial: Union[bool, Sequence[str], Set[str]] = None*, *unknown: 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)` duple. A `ValidationError` is raised if invalid data are passed.

loads(*json_data: str*, ***, *many: bool = None*, *partial: Union[bool, Sequence[str], Set[str]] = None*, *unknown: 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)` duple. 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

validate(*data: Mapping*, ***, *many: bool = None*, *partial: 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.utils.marshmallow_schema.**RoleSchema**(*, *only: Union[Sequence[str], Set[str]] = None*, *exclude: Union[Sequence[str], Set[str]] = ()*, *many: bool = False*, *context: 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: str = None*)

class Meta

fields = ('id', 'name', 'description', 'label', 'created_at', 'updated_at', 'delete

ordered = True

OPTIONS_CLASS
alias of `marshmallow.schema.SchemaOpts`

TYPE_MAPPING = {<class 'str'>: <class 'marshmallow.fields.String'>, <class 'bytes'>:

_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.Timestamp(default=<marshmallow.missing>, at

_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[_T, List[_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: bool = None*, *partial: Union[bool, Sequence[str], Set[str]] = None*, *unknown: 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[_T, Iterable[_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

dump(*obj: Any*, *, *many: 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 A dict of serialized data

Return type dict

New in version 1.0.0.

Changed in version 3.0.0b7: This method returns the serialized data rather than a `(data, errors)` duple. 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: 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

Return type str

New in version 1.0.0.

Changed in version 3.0.0b7: This method returns the serialized data rather than a `(data, errors)` duple. A `ValidationError` is raised if `obj` is invalid.

error_messages = {}

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: bool = None*, *partial: Union[bool, Sequence[str], Set[str]] = None*, *unknown: 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)` duple. A `ValidationError` is raised if invalid data are passed.

loads(*json_data: str*, ***, *many: bool = None*, *partial: Union[bool, Sequence[str], Set[str]] = None*, *unknown: 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)` duple. 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

static valid_request_role(*data: dict*) → dict

validate(*data: Mapping*, ***, *many: bool = None*, *partial: 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.utils.marshmallow_schema.**SearchSchema**(*, *only: Union[Sequence[str], Set[str]] = None*, *exclude: Union[Sequence[str], Set[str]] = ()*, *many: bool = False*, *context: 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: 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.
- **`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'>:

_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(default=<marshmallow.missing>, a

_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[_T, List[_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: bool = None*, *partial: Union[bool, Sequence[str], Set[str]] = None*, *unknown: 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[_T, Iterable[_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

dump(*obj: Any*, ***, *many: 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 A dict of serialized data

Return type dict

New in version 1.0.0.

Changed in version 3.0.0b7: This method returns the serialized data rather than a `(data, errors)` duple. 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: 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

Return type str

New in version 1.0.0.

Changed in version 3.0.0b7: This method returns the serialized data rather than a `(data, errors)` duple. A `ValidationError` is raised if `obj` is invalid.

error_messages = {}

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: bool = None*, *partial: Union[bool, Sequence[str], Set[str]] = None*, *unknown: 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)` duple. A `ValidationError` is raised if invalid data are passed.

loads(*json_data: str*, ***, *many: bool = None*, *partial: Union[bool, Sequence[str], Set[str]] = None*, *unknown: 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)` duple. 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

validate(*data: Mapping*, ***, *many: bool = None*, *partial: 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.utils.marshmallow_schema.**Timestamp**(**, default: Any = <marshmallow.missing>, missing: Any = <marshmallow.missing>, data_key: str = None, attribute: str = None, validate: Union[Callable[[Any], Any], Iterable[Callable[[Any], Any]]] = None, required: bool = False, allow_none: bool = None, load_only: bool = False, dump_only: bool = False, error_messages: Dict[str, str] = None, **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*) – Parent schema.

_creation_index = 73

_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.

_validate_missing(*value*)

Validate missing values. Raise a `ValidationError` if *value* should be considered missing.

property context

The context dictionary for the parent `Schema`.

default_error_messages = {'null': 'Field may not be null.', 'required': 'Missing dat

deserialize(*value: Any*, *attr: str = None*, *data: 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`.

name = None

parent = None

property root
Reference to the *Schema* that this field belongs to even if it is buried in a container field (e.g. *List*). Return *None* for unbound fields.

serialize(*attr: str*, *obj: Any*, *accessor: 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.utils.marshmallow_schema.**UserSchema**(*, *only: Union[Sequence[str], Set[str]] = None*, *exclude: Union[Sequence[str], Set[str]] = ()*, *many: bool = False*, *context: 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: str = None*)

class Meta

fields = ('id', 'name', 'last_name', 'email', 'genre', 'birth_date', 'active', 'cre

ordered = True

OPTIONS_CLASS
alias of `marshmallow.schema.SchemaOpts`

TYPE_MAPPING = {<class 'str'>: <class 'marshmallow.fields.String'>, <class 'bytes'>:

_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(default=<marshmallow.missing>, attribut

_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[_T, List[_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: bool = None*, *partial: Union[bool, Sequence[str], Set[str]] = None*, *unknown: 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[_T, Iterable[_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

dump(*obj: Any*, ***, *many: 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 A dict of serialized data

Return type dict

New in version 1.0.0.

Changed in version 3.0.0b7: This method returns the serialized data rather than a `(data, errors)` duple. 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: 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

Return type str

New in version 1.0.0.

Changed in version 3.0.0b7: This method returns the serialized data rather than a `(data, errors)` duple. A `ValidationError` is raised if `obj` is invalid.

error_messages = {}

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: bool = None*, *partial: Union[bool, Sequence[str], Set[str]] = None*, *unknown: 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)` duple. A `ValidationError` is raised if invalid data are passed.

loads(*json_data: str*, ***, *many: bool = None*, *partial: Union[bool, Sequence[str], Set[str]] = None*, *unknown: 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)` duple. 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

static valid_request_user(*data: dict*) → dict

validate(*data: Mapping*, *, *many: bool = None*, *partial: 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_credentials(*data: dict*) → dict

static validate_email(*data: dict*) → dict

static validate_password(*password: str*) → None

class app.utils.marshmallow_schema.**_SearchOrderSchema**(*, *only: Union[Sequence[str], Set[str]] = None*, *exclude: Union[Sequence[str], Set[str]] = ()*, *many: bool = False*, *context: 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: 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.
- **`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'>:`

`_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(default=<marshmallow.missing>, attril`

`_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[_T, List[_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: bool = None*, *partial: Union[bool, Sequence[str], Set[str]] = None*, *unknown: 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[_T, Iterable[_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

dump(*obj: Any*, *, *many: 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 A dict of serialized data

Return type dict

New in version 1.0.0.

Changed in version 3.0.0b7: This method returns the serialized data rather than a `(data, errors)` duple. 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: 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

Return type str

New in version 1.0.0.

Changed in version 3.0.0b7: This method returns the serialized data rather than a `(data, errors)` duple. A `ValidationError` is raised if `obj` is invalid.

error_messages = {}

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: bool = None*, *partial: Union[bool, Sequence[str], Set[str]] = None*, *unknown: 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)` duple. A `ValidationError` is raised if invalid data are passed.

loads(*json_data: str*, ***, *many: bool = None*, *partial: Union[bool, Sequence[str], Set[str]] = None*, *unknown: 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)` duple. 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

validate(*data: Mapping*, ***, *many: bool = None*, *partial: 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.utils.marshmallow_schema.**_SearchValueSchema**(*, *only: Union[Sequence[str], Set[str]] = None*, *exclude: Union[Sequence[str], Set[str]] = ()*, *many: bool = False*, *context: 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: 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.
- **`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'>:

_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(default=<marshmallow.missing>, attri

_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[_T, List[_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: bool = None*, *partial: Union[bool, Sequence[str], Set[str]] = None*, *unknown: 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[_T, Iterable[_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

dump(*obj: Any*, ***, *many: 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 A dict of serialized data

Return type dict

New in version 1.0.0.

Changed in version 3.0.0b7: This method returns the serialized data rather than a `(data, errors)` duple. 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: 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

Return type str

New in version 1.0.0.

Changed in version 3.0.0b7: This method returns the serialized data rather than a `(data, errors)` duple. A `ValidationError` is raised if `obj` is invalid.

error_messages = {}

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: bool = None*, *partial: Union[bool, Sequence[str], Set[str]] = None*, *unknown: 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)` duple. A `ValidationError` is raised if invalid data are passed.

loads(*json_data: str*, ***, *many: bool = None*, *partial: Union[bool, Sequence[str], Set[str]] = None*, *unknown: 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)` duple. 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

validate(*data: Mapping*, ***, *many: bool = None*, *partial: 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.utils.swagger_models

Description

Modules

app.utils.swagger_models.auth
app.utils.swagger_models.document
app.utils.swagger_models.role
app.utils.swagger_models.user

app.utils.swagger_models.auth

Description

app.utils.swagger_models.document

Description

app.utils.swagger_models.role

Description

app.utils.swagger_models.user

Description

Functions

class_for_name(module_name, class_name)	
create_search_query(db_model, query[, data])	
find_longest_word(word_list)	
get_request_query_fields(db_model[, ...])	
ignore_keys(data, exclude)	
pos_to_char(pos)	
to_readable(obj)	

app.utils.class_for_name

app.utils.**class_for_name**(*module_name: str*, *class_name: str*) → any

app.utils.create_search_query

app.utils.**create_search_query**(*db_model: Type[peewee.Model]*, *query: peewee.ModelSelect*, *data: dict = None*) → peewee.ModelSelect

app.utils.find_longest_word

app.utils.**find_longest_word**(*word_list: list*) → str

app.utils.get_request_query_fields

app.utils.**get_request_query_fields**(*db_model: Type[peewee.Model]*, *request_data=None*) → tuple

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

Attributes

STRING_QUERY_OPERATORS	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.STRING_QUERY_OPERATORS

app.utils.**STRING_QUERY_OPERATORS = ['eq', 'ne', 'contains', 'ncontains', 'startswith', 'enc**
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]

Exceptions

FileEmptyError

app.utils.FileEmptyError

exception app.utils.**FileEmptyError**

exception app.utils.**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.utils.**STRING_QUERY_OPERATORS = ['eq', 'ne', 'contains', 'ncontains', 'startswith', 'enc**
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.**_build_clause_operators**(*field: peewee.Field*, *field_operator: str*, *field_value*) → tuple

app.utils.**_build_order_by**(*db_model: Type[peewee.Model]*, *request_data: dict*) → list

app.utils.**_build_query_clause**(*field: peewee.Field*, *field_operator: str*, *field_value*)

app.utils.**_build_string_clause**(*field: peewee.Field*, *field_operator: str*, *field_value*) → tuple

app.utils.**class_for_name**(*module_name: str*, *class_name: str*) → any

app.utils.**create_search_query**(*db_model: Type[peewee.Model]*, *query: peewee.ModelSelect*, *data: dict = None*) → peewee.ModelSelect

app.utils.**find_longest_word**(*word_list: list*) → str

app.utils.**get_request_query_fields**(*db_model: Type[peewee.Model]*, *request_data=None*) → tuple

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

create_app(env_config)	Builds an application based on environment configuration.

2.1.7 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_app**(*app: flask.app.Flask*) → None
Call the method 'init_app' to register the extensions in the Flask object passed as parameter.

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

database.factories	Package contains factories modules.
database.migrations	
database.seeds	

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

Factory(model_name, records)	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
```

(continues on next page)

(continued from previous page)

```
'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

Factory.__init__(model_name[, records])	Register as many factories as given records.
Factory.make([params, to_dict, exclude])	Create instances of database model with fake data.
Factory.save([params])	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: dict = None*, *to_dict: bool = False*, *exclude: 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: 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>]
```

make (*params: dict = None*, *to_dict: bool = False*, *exclude: 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: 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.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__()	Initialize self.
AddGenreColumnOnUserTable.down()	
AddGenreColumnOnUserTable.up()	

database.migrations.aaa_add_genre_column_on_user_table.AddGenreColumnOnUserTable.__init__

AddGenreColumnOnUserTable.**__init__**()
Initialize self. See help(type(self)) for accurate signature.

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.**AddCreatedByColumnOnUserT**
Bases: object

Methods

AddCreatedByColumnOnUserTable.__init__()	Initialize self.
AddCreatedByColumnOnUserTable.down()	
AddCreatedByColumnOnUserTable.up()	

database.migrations.aab_add_created_by_column_on_user_table.AddCreatedByColumnOnUserTable.__

AddCreatedByColumnOnUserTable.**__init__**()
Initialize self. See help(type(self)) for accurate signature.

database.migrations.aab_add_created_by_column_on_user_table.AddCreatedByColumnOnUserTable.do

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.**AddCreatedByColumnOnUserT**

_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__()	Initialize self.
CreateDocumentsTable.down()	
CreateDocumentsTable.up()	

database.migrations.aac_create_documents_table.CreateDocumentsTable.__init__

CreateDocumentsTable.**__init__**()
Initialize self. See help(type(self)) for accurate signature.

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__()	Initialize self.
CreateUserRolesTable.down()	
CreateUserRolesTable.up()	

database.migrations.aad_create_user_roles_table.CreateUserRolesTable.__init__

CreateUserRolesTable.**__init__**()
Initialize self. See help(type(self)) for accurate signature.

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 _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*)

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: list = None*, *include: list = None*, *sort_order: 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>

classmethod raw(*sql*, **params*)

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__()	Initialize self.
RemoveRoleSlugColumn.down()	
RemoveRoleSlugColumn.up()	

database.migrations.aaf_remove_role_slug_column.RemoveRoleSlugColumn.__init__

RemoveRoleSlugColumn.**__init__**()
Initialize self. See help(type(self)) for accurate signature.

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 _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*)
Add one or more permissions to role.

Parameters permissions – a set, list, or single string.

Caller must commit to DB.

New in version 3.3.0.

Deprecated since version 3.4.4: Use UserDatastore.remove_permissions_from_role()

classmethod alias(*alias=None*)

classmethod bind(*database*, *bind_refs=True*, *bind_backrefs=True*)

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: list = None*, *include: list = None*, *sort_order: list = None*) → set

get_id()

classmethod get_or_create(***kwargs*)

classmethod get_or_none(**query*, ***filters*)

get_permissions()

Return set of permissions associated with role.

Either takes a comma separated string of permissions or an interable of strings if permissions are in their own table.

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()

classmethod raw(*sql*, **params*)

remove_permissions(*permissions*)

Remove one or more permissions from role.

Parameters **permissions** – a set, list, or single string.

Caller must commit to DB.

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()

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)	Initialize self.
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)	

continues on next page

Table 125 – continued from previous page

Migration.get_by_id(pk)
Migration.get_id()
Migration.get_or_create(**kwargs)
Migration.get_or_none(*query, **filters)
Migration.index(*fields, **kwargs)
Migration.insert([_Model__data])
Migration.insert_from(query, fields)
Migration.insert_many(rows[, fields])
Migration.is_alias()
Migration.is_dirty()
Migration.noop()
Migration.raw(sql, *params)
Migration.replace([_Model__data])
Migration.replace_many(rows[, fields])
Migration.save([force_insert, only])
Migration.select(*fields)
Migration.set_by_id(key, value)
Migration.table_exists()
Migration.truncate_table(**options)
Migration.unwrap()
Migration.update([_Model__data])
Migration.validate_model()

database.migrations.Migration.__init__

Migration.**__init__**(**args*, ***kwargs*)

Initialize self. See help(type(self)) for accurate signature.

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*)

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 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*)

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(**kwargs)

database.seeds.document_seeder.DocumentSeeder

class database.seeds.document_seeder.**DocumentSeeder**(***kwargs*)
Bases: object

Attributes

DocumentSeeder.name

database.seeds.document_seeder.DocumentSeeder.name

DocumentSeeder.**name = 'DocumentSeeder'**

Methods

DocumentSeeder.__init__([rows])	Initialize self.

database.seeds.document_seeder.DocumentSeeder.__init__

DocumentSeeder.**__init__**(*rows: int = 30*)
Initialize self. See help(type(self)) for accurate signature.

class database.seeds.document_seeder.**DocumentSeeder**(***kwargs*)

name = 'DocumentSeeder'

database.seeds.role_seeder

Description

Classes

RoleSeeder(**kwargs)

database.seeds.role_seeder.RoleSeeder

class database.seeds.role_seeder.**RoleSeeder**(***kwargs*)
Bases: object

Attributes

RoleSeeder.name

database.seeds.role_seeder.RoleSeeder.name

RoleSeeder.**name = 'RoleSeeder'**

Methods

RoleSeeder.__init__()	Initialize self.

database.seeds.role_seeder.RoleSeeder.__init__

RoleSeeder.**__init__**()
Initialize self. See help(type(self)) for accurate signature.

class database.seeds.role_seeder.**RoleSeeder**(***kwargs*)

static _create_admin_role() → None

static _create_team_leader() → None

static _create_worker_role() → None

name = 'RoleSeeder'

database.seeds.user_seeder

Description

Classes

UserSeeder(**kwargs)

database.seeds.user_seeder.UserSeeder

class database.seeds.user_seeder.**UserSeeder**(***kwargs*)
Bases: object

Attributes

UserSeeder.name

database.seeds.user_seeder.UserSeeder.name

UserSeeder.**name = 'UserSeeder'**

Methods

UserSeeder.__init__([rows])	Initialize self.

database.seeds.user_seeder.UserSeeder.__init__

UserSeeder.**__init__**(*rows: int = 30*)
Initialize self. See help(type(self)) for accurate signature.

class database.seeds.user_seeder.**UserSeeder**(***kwargs*)

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**(*fnc*)

database.**init_database**() → None

database.**seed_actions**(*fnc*)

2.3 tests

Description

Package for testing the application.

The tests package storages the application's tests that them are executed for ensuring the proper behaviour of the application.

You can get report coverage stadistics 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 -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 stadistics 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
```

Modules

tests.blueprints	Package for testing blueprints.
tests.celery	Package for testing Celery tasks.
tests.conftest	Module for configuring Pytest.
tests.test_config	Module for testing Config module.
tests.test_db	Module for testing database.
tests.test_mail	Module for testing mail.

2.3.1 tests.blueprints

Description

Package for testing blueprints.

Modules

tests.blueprints.test_auth	Module for testing auth blueprint.
tests.blueprints.test_base	Module for testing base blueprint.
tests.blueprints.test_documents	Module for testing documents blueprint.
tests.blueprints.test_roles	Module for testing roles blueprint.
tests.blueprints.test_users	Module for testing users blueprint.

tests.blueprints.test_auth

Description

Module for testing auth blueprint.

Functions

test_request_reset_password(client)
test_reset_password(client)
test_user_login(client)
test_user_logout(client, auth_header)
test_validate_reset_password(client, app)

tests.blueprints.test_auth.test_request_reset_password

tests.blueprints.test_auth.**test_request_reset_password**(*client: flask.testing.FlaskClient*)

tests.blueprints.test_auth.test_reset_password

tests.blueprints.test_auth.**test_reset_password**(*client: flask.testing.FlaskClient*)

tests.blueprints.test_auth.test_user_login

tests.blueprints.test_auth.**test_user_login**(*client: flask.testing.FlaskClient*)

tests.blueprints.test_auth.test_user_logout

tests.blueprints.test_auth.**test_user_logout**(*client: flask.testing.FlaskClient*, *auth_header: any*)

tests.blueprints.test_auth.test_validate_reset_password

tests.blueprints.test_auth.**test_validate_reset_password**(*client: flask.testing.FlaskClient*, *app: flask.app.Flask*)

tests.blueprints.test_auth.**test_request_reset_password**(*client: flask.testing.FlaskClient*)

tests.blueprints.test_auth.**test_reset_password**(*client: flask.testing.FlaskClient*)

tests.blueprints.test_auth.**test_user_login**(*client: flask.testing.FlaskClient*)

tests.blueprints.test_auth.**test_user_logout**(*client: flask.testing.FlaskClient*, *auth_header: any*)

tests.blueprints.test_auth.**test_validate_reset_password**(*client: flask.testing.FlaskClient*, *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: flask.testing.FlaskClient*)

tests.blueprints.test_base.**test_welcome_api**(*client: flask.testing.FlaskClient*)

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: flask.testing.FlaskClient*, *auth_header: any*)

tests.blueprints.test_documents.test_get_document_data

tests.blueprints.test_documents.**test_get_document_data**(*client: flask.testing.FlaskClient*, *auth_header: any*)

tests.blueprints.test_documents.test_get_document_file

tests.blueprints.test_documents.**test_get_document_file**(*client: flask.testing.FlaskClient*, *auth_header: any*)

tests.blueprints.test_documents.test_save_document

tests.blueprints.test_documents.**test_save_document**(*client: flask.testing.FlaskClient*, *auth_header: any*)

tests.blueprints.test_documents.test_search_document

tests.blueprints.test_documents.**test_search_document**(*client: flask.testing.FlaskClient*, *auth_header: any*)

tests.blueprints.test_documents.test_update_document

tests.blueprints.test_documents.**test_update_document**(*client: flask.testing.FlaskClient*, *auth_header: any*)

tests.blueprints.test_documents.**test_delete_document**(*client: flask.testing.FlaskClient*, *auth_header: any*)

tests.blueprints.test_documents.**test_get_document_data**(*client: flask.testing.FlaskClient*, *auth_header: any*)

tests.blueprints.test_documents.**test_get_document_file**(*client: flask.testing.FlaskClient*, *auth_header: any*)

tests.blueprints.test_documents.**test_save_document**(*client: flask.testing.FlaskClient*, *auth_header: any*)

tests.blueprints.test_documents.**test_search_document**(*client: flask.testing.FlaskClient*, *auth_header: any*)

tests.blueprints.test_documents.**test_update_document**(*client: flask.testing.FlaskClient*, *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: flask.testing.FlaskClient*, *auth_header: any*)

tests.blueprints.test_roles.test_get_role_endpoint

tests.blueprints.test_roles.**test_get_role_endpoint**(*client: flask.testing.FlaskClient*, *auth_header: any*)

tests.blueprints.test_roles.test_save_role_endpoint

tests.blueprints.test_roles.**test_save_role_endpoint**(*client: flask.testing.FlaskClient*, *auth_header: any*, *factory: any*)

tests.blueprints.test_roles.test_search_roles_endpoint

tests.blueprints.test_roles.**test_search_roles_endpoint**(*client: flask.testing.FlaskClient*, *auth_header: any*)

tests.blueprints.test_roles.test_update_role_endpoint

tests.blueprints.test_roles.**test_update_role_endpoint**(*client: flask.testing.FlaskClient*, *auth_header: any*, *factory: any*)

tests.blueprints.test_roles.**test_delete_role_endpoint**(*client: flask.testing.FlaskClient*, *auth_header: any*)

tests.blueprints.test_roles.**test_get_role_endpoint**(*client: flask.testing.FlaskClient*, *auth_header: any*)

tests.blueprints.test_roles.**test_save_role_endpoint**(*client: flask.testing.FlaskClient*, *auth_header: any*, *factory: any*)

tests.blueprints.test_roles.**test_search_roles_endpoint**(*client: flask.testing.FlaskClient*, *auth_header: any*)

tests.blueprints.test_roles.**test_update_role_endpoint**(*client: flask.testing.FlaskClient*, *auth_header: any*, *factory: any*)

tests.blueprints.test_users

Description

Module for testing users blueprint.

Functions

test_delete_user_endpoint(client, auth_header)	
test_export_excel_endpoint(client, auth_header)	
test_export_word_endpoint(client, auth_header)	
test_get_user_endpoint(client, auth_header)	
test_save_user_endpoint(client, auth_header, ...)	
test_search_users_endpoint(client, auth_header)	
test_update_user_endpoint(client, ...)	

tests.blueprints.test_users.test_delete_user_endpoint

tests.blueprints.test_users.**test_delete_user_endpoint**(*client: flask.testing.FlaskClient*, *auth_header: any*)

tests.blueprints.test_users.test_export_excel_endpoint

tests.blueprints.test_users.**test_export_excel_endpoint**(*client: flask.testing.FlaskClient*, *auth_header: any*)

tests.blueprints.test_users.test_export_word_endpoint

tests.blueprints.test_users.**test_export_word_endpoint**(*client: flask.testing.FlaskClient*, *auth_header: any*)

tests.blueprints.test_users.test_get_user_endpoint

tests.blueprints.test_users.**test_get_user_endpoint**(*client: flask.testing.FlaskClient*, *auth_header: any*)

tests.blueprints.test_users.test_save_user_endpoint

tests.blueprints.test_users.**test_save_user_endpoint**(*client: flask.testing.FlaskClient*, *auth_header: any*, *factory: any*)

tests.blueprints.test_users.test_search_users_endpoint

tests.blueprints.test_users.**test_search_users_endpoint**(*client: flask.testing.FlaskClient*, *auth_header: any*)

tests.blueprints.test_users.test_update_user_endpoint

tests.blueprints.test_users.**test_update_user_endpoint**(*client: flask.testing.FlaskClient*, *auth_header: any*, *factory: any*)

tests.blueprints.test_users.**test_delete_user_endpoint**(*client: flask.testing.FlaskClient*, *auth_header: any*)

tests.blueprints.test_users.**test_export_excel_endpoint**(*client: flask.testing.FlaskClient*, *auth_header: any*)

tests.blueprints.test_users.**test_export_word_endpoint**(*client: flask.testing.FlaskClient*, *auth_header: any*)

tests.blueprints.test_users.**test_get_user_endpoint**(*client: flask.testing.FlaskClient*, *auth_header: any*)

tests.blueprints.test_users.**test_save_user_endpoint**(*client: flask.testing.FlaskClient*, *auth_header: any*, *factory: any*)

tests.blueprints.test_users.**test_search_users_endpoint**(*client: flask.testing.FlaskClient*, *auth_header: any*)

tests.blueprints.test_users.**test_update_user_endpoint**(*client: flask.testing.FlaskClient*, *auth_header: any*, *factory: any*)

2.3.2 tests.celery

Description

Package for testing Celery tasks.

Modules

tests.celery.test_excel	Module for testing excel module.
tests.celery.test_tasks	Module for testing task module.
tests.celery.test_word	Module for testing word module.

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)

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_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_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

app()	Create an app with testing environment.
auth_header(app, client)	Create an auth header from a given user that can be added to an http requests.
client(app)	Create a test client for making http requests.
factory(app)	Create a Factory from a database model.
runner(app)	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: flask.testing.FlaskClient*)
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: flask.testing.FlaskClient*)
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.test_config

Description

Module for testing Config module.

Functions

test_config()	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.

2.3.5 tests.test_db

Description

Module for testing database.

Functions

test_get_close_db()	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.

2.3.6 tests.test_mail

Description

Module for testing mail.

Functions

test_mail_record_messages(app)	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

2.4 config

Description

Module loads the application's configuration.

The extension and custom configurations are defined here.

Classes

Config()	Default configuration options.
DevConfig()	Development configuration options.
Meta(name, bases, dict)	Metaclass for updating Config options.
ProdConfig()	Production configuration options.
TestConfig()	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.ERROR_404_HELP
Config.FLASK_RESTFUL_PREFIX
Config.HOME
Config.LOGIN_DISABLED
Config.LOG_DIRECTORY
Config.MAIL_PASSWORD

continues on next page

Table 155 – continued from previous page

Config.MAIL_PORT
Config.MAIL_SERVER
Config.MAIL_USERNAME
Config.MAIL_USE_SSL
Config.MAIL_USE_TLS
Config.RESET_TOKEN_EXPIRES
Config.RESTX_MASK_SWAGGER
Config.ROOT_DIRECTORY
Config.SECRET_KEY
Config.SECURITY_PASSWORD_HASH
Config.SECURITY_PASSWORD_LENGTH_MIN
Config.SECURITY_PASSWORD_SALT
Config.SECURITY_TOKEN_AUTHENTICATION_HEADER
Config.SECURITY_TOKEN_MAX_AGE
Config.SERVER_NAME
Config.STORAGE_DIRECTORY
Config.SWAGGER_API_URL
Config.SWAGGER_URL
Config.TESTING
Config.TEST_USER_EMAIL
Config.TEST_USER_PASSWORD
Config.accept_content
Config.broker_url
Config.enable_utc
Config.include
Config.result_backend
Config.result_expires
Config.result_extended
Config.result_serializer
Config.task_default_rate_limit
Config.task_serializer
Config.task_track_started
Config.timezone
Config.worker_log_format
Config.worker_task_log_format

config.Config.ALLOWED_CONTENT_TYPES

Config.**ALLOWED_CONTENT_TYPES = {'application/json', 'application/octet-stream', 'mu**

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':**

config.Config.DEBUG

Config.**DEBUG = False**

config.Config.DEVELOPMENT

Config.**DEVELOPMENT = False**

config.Config.ERROR_404_HELP

Config.**ERROR_404_HELP = 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/**

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.RESET_TOKEN_EXPIRES

Config.**RESET_TOKEN_EXPIRES = 86400**

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**

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-**

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_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 - %(messa**

config.Config.worker_task_log_format

Config.**worker_task_log_format = '%(asctime)s - %(levelname)s - %(processName)s - %**

Methods

Config.__init__()	Initialize self.

config.Config.__init__

Config.**__init__**()
Initialize self. See help(type(self)) for accurate signature.

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.ERROR_404_HELP
DevConfig.FLASK_RESTFUL_PREFIX
DevConfig.HOME

continues on next page

Table 157 – continued from previous page

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.RESET_TOKEN_EXPIRES
DevConfig.RESTX_MASK_SWAGGER
DevConfig.ROOT_DIRECTORY
DevConfig.SECRET_KEY
DevConfig.SECURITY_PASSWORD_HASH
DevConfig.SECURITY_PASSWORD_LENGTH_MIN
DevConfig.SECURITY_PASSWORD_SALT
DevConfig.SECURITY_TOKEN_AUTHENTICATION_HEADER
DevConfig.SECURITY_TOKEN_MAX_AGE
DevConfig.SERVER_NAME
DevConfig.STORAGE_DIRECTORY
DevConfig.SWAGGER_API_URL
DevConfig.SWAGGER_URL
DevConfig.TESTING
DevConfig.TEST_USER_EMAIL
DevConfig.TEST_USER_PASSWORD
DevConfig.accept_content
DevConfig.broker_url
DevConfig.enable_utc
DevConfig.include
DevConfig.result_backend
DevConfig.result_expires
DevConfig.result_extended
DevConfig.result_serializer
DevConfig.task_default_rate_limit
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',**

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':**

config.DevConfig.DEBUG

DevConfig.**DEBUG = True**

config.DevConfig.DEVELOPMENT

DevConfig.**DEVELOPMENT = True**

config.DevConfig.ERROR_404_HELP

DevConfig.**ERROR_404_HELP = False**

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-a**

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.RESET_TOKEN_EXPIRES

DevConfig.**RESET_TOKEN_EXPIRES = 86400**

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-**

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/fla**

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_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 - %(me
```

config.DevConfig.worker_task_log_format

```
DevConfig.worker_task_log_format = '%(asctime)s - %(levelname)s - %(processName)s -
```

Methods

DevConfig.__init__()	Initialize self.

config.DevConfig.__init__

DevConfig.**__init__**()
Initialize self. See help(type(self)) for accurate signature.

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)	Initialize self.
Meta.mro()	Return a type's method resolution order.

config.Meta.__init__

Meta.**__init__**(**args*, ***kwargs*)
Initialize self. See help(type(self)) for accurate signature.

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.ERROR_404_HELP
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

continues on next page

Table 160 – continued from previous page

ProdConfig.RESET_TOKEN_EXPIRES
ProdConfig.RESTX_MASK_SWAGGER
ProdConfig.ROOT_DIRECTORY
ProdConfig.SECRET_KEY
ProdConfig.SECURITY_PASSWORD_HASH
ProdConfig.SECURITY_PASSWORD_LENGTH_MIN
ProdConfig.SECURITY_PASSWORD_SALT
ProdConfig.SECURITY_TOKEN_AUTHENTICATION_HEADER
ProdConfig.SECURITY_TOKEN_MAX_AGE
ProdConfig.SERVER_NAME
ProdConfig.STORAGE_DIRECTORY
ProdConfig.SWAGGER_API_URL
ProdConfig.SWAGGER_URL
ProdConfig.TESTING
ProdConfig.TEST_USER_EMAIL
ProdConfig.TEST_USER_PASSWORD
ProdConfig.accept_content
ProdConfig.broker_url
ProdConfig.enable_utc
ProdConfig.include
ProdConfig.result_backend
ProdConfig.result_expires
ProdConfig.result_extended
ProdConfig.result_serializer
ProdConfig.task_default_rate_limit
ProdConfig.task_serializer
ProdConfig.task_track_started
ProdConfig.timezone
ProdConfig.worker_log_format
ProdConfig.worker_task_log_format

config.ProdConfig.ALLOWED_CONTENT_TYPES

ProdConfig.**ALLOWED_CONTENT_TYPES = {'application/json', 'application/octet-stream',**

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'**

config.ProdConfig.DEBUG

ProdConfig.**DEBUG = False**

config.ProdConfig.DEVELOPMENT

ProdConfig.**DEVELOPMENT = False**

config.ProdConfig.ERROR_404_HELP

ProdConfig.**ERROR_404_HELP = 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-**

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.RESET_TOKEN_EXPIRES

ProdConfig.**RESET_TOKEN_EXPIRES = 86400**

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**

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/fl**

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_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 - %(n**

config.ProdConfig.worker_task_log_format

ProdConfig.**worker_task_log_format = '%(asctime)s - %(levelname)s - %(processName)s**

Methods

ProdConfig.__init__()	Initialize self.

config.ProdConfig.__init__

ProdConfig.**__init__**()
Initialize self. See help(type(self)) for accurate signature.

2.4.5 config.TestConfig

class config.**TestConfig**
Bases: *config.Config*

Testing configuration options.

Attributes

TestConfig.ALLOWED_CONTENT_TYPES
TestConfig.ALLOWED_MIME_TYPES
TestConfig.DATABASE
TestConfig.DEBUG
TestConfig.DEVELOPMENT
TestConfig.ERROR_404_HELP
TestConfig.FLASK_RESTFUL_PREFIX

continues on next page

Table 162 – continued from previous page

TestConfig.HOME
TestConfig.LOGIN_DISABLED
TestConfig.LOG_DIRECTORY
TestConfig.MAIL_PASSWORD
TestConfig.MAIL_PORT
TestConfig.MAIL_SERVER
TestConfig.MAIL_USERNAME
TestConfig.MAIL_USE_SSL
TestConfig.MAIL_USE_TLS
TestConfig.RESET_TOKEN_EXPIRES
TestConfig.RESTX_MASK_SWAGGER
TestConfig.ROOT_DIRECTORY
TestConfig.SECRET_KEY
TestConfig. *SECURITY_PASSWORD_HASH*
TestConfig. *SECURITY_PASSWORD_LENGTH_MIN*
TestConfig. *SECURITY_PASSWORD_SALT*
TestConfig. *SECURITY_TOKEN_AUTHENTICATION_HEADER*
TestConfig. *SECURITY_TOKEN_MAX_AGE*
TestConfig.SERVER_NAME
TestConfig.STORAGE_DIRECTORY
TestConfig.SWAGGER_API_URL
TestConfig.SWAGGER_URL
TestConfig.TESTING
TestConfig.TEST_USER_EMAIL
TestConfig.TEST_USER_PASSWORD
TestConfig.accept_content
TestConfig.broker_url
TestConfig.enable_utc
TestConfig.include
TestConfig.result_backend
TestConfig.result_expires
TestConfig.result_extended
TestConfig.result_serializer
TestConfig. *task_default_rate_limit*
TestConfig.task_serializer
TestConfig.task_track_started
TestConfig.timezone
TestConfig.worker_log_format
TestConfig. *worker_task_log_format*

config.TestConfig.ALLOWED_CONTENT_TYPES

TestConfig.**ALLOWED_CONTENT_TYPES = {'application/json', 'application/octet-stream',**

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', 'pra**

config.TestConfig.DEBUG

TestConfig.**DEBUG = True**

config.TestConfig.DEVELOPMENT

TestConfig.**DEVELOPMENT = True**

config.TestConfig.ERROR_404_HELP

TestConfig.**ERROR_404_HELP = False**

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-**

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.RESET_TOKEN_EXPIRES

TestConfig.**RESET_TOKEN_EXPIRES = 86400**

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**

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/fl**

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_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 - %(m**

config.TestConfig.worker_task_log_format

TestConfig.**worker_task_log_format = '%(asctime)s - %(levelname)s - %(processName)s**

Methods

TestConfig.__init__()	Initialize self.

config.TestConfig.__init__

TestConfig.**__init__**()
Initialize self. See help(type(self)) for accurate signature.

class config.**Config**
Default configuration options.

ALLOWED_CONTENT_TYPES = {'application/json', 'application/octet-stream', 'multipart/fo

ALLOWED_MIME_TYPES = {'application/pdf', 'application/vnd.ms-excel'}

DATABASE = {'engine': 'peewee.SqliteDatabase', 'name': None, 'pragmas': {'cache_siz

DEBUG = False

DEVELOPMENT = False

ERROR_404_HELP = False

FLASK_RESTFUL_PREFIX = '/api'

HOME = '/home/docs'

LOGIN_DISABLED = False

```
LOG_DIRECTORY = '/home/docs/checkouts/readthedocs.org/user_builds/flask-api/checkouts/
MAIL_PASSWORD = None
MAIL_PORT = None
MAIL_SERVER = None
MAIL_USERNAME = None
MAIL_USE_SSL = False
MAIL_USE_TLS = True
RESET_TOKEN_EXPIRES = 86400
RESTX_MASK_SWAGGER = False
ROOT_DIRECTORY = '/home/docs/checkouts/readthedocs.org/user_builds/flask-api/checkouts
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/checko
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_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)
```

class config.**DevConfig**

Development configuration options.

```
    ALLOWED_CONTENT_TYPES = {'application/json', 'application/octet-stream', 'multipart/fo
    ALLOWED_MIME_TYPES = {'application/pdf', 'application/vnd.ms-excel'}
    DATABASE = {'engine':  'peewee.SqliteDatabase', 'name':  None, 'pragmas':  {'cache_siz
    DEBUG = True
    DEVELOPMENT = True
    ERROR_404_HELP = False
    FLASK_RESTFUL_PREFIX = '/api'
    HOME = '/home/docs'
    LOGIN_DISABLED = False
    LOG_DIRECTORY = '/home/docs/checkouts/readthedocs.org/user_builds/flask-api/checkouts/
    MAIL_PASSWORD = None
    MAIL_PORT = None
    MAIL_SERVER = None
    MAIL_USERNAME = None
    MAIL_USE_SSL = False
    MAIL_USE_TLS = True
    RESET_TOKEN_EXPIRES = 86400
    RESTX_MASK_SWAGGER = False
    ROOT_DIRECTORY = '/home/docs/checkouts/readthedocs.org/user_builds/flask-api/checkouts
    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/checko
    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_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)

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/fo

ALLOWED_MIME_TYPES = {'application/pdf', 'application/vnd.ms-excel'}

DATABASE = {'engine': 'peewee.SqliteDatabase', 'name': None, 'pragmas': {'cache_siz

DEBUG = False

DEVELOPMENT = False

ERROR_404_HELP = False

FLASK_RESTFUL_PREFIX = '/api'

HOME = '/home/docs'

LOGIN_DISABLED = False

LOG_DIRECTORY = '/home/docs/checkouts/readthedocs.org/user_builds/flask-api/checkouts/

MAIL_PASSWORD = None

MAIL_PORT = None

MAIL_SERVER = None

```
    MAIL_USERNAME = None
    MAIL_USE_SSL = False
    MAIL_USE_TLS = True
    RESET_TOKEN_EXPIRES = 86400
    RESTX_MASK_SWAGGER = False
    ROOT_DIRECTORY = '/home/docs/checkouts/readthedocs.org/user_builds/flask-api/checkouts
    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/checko
    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_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)
```

class config.**TestConfig**

Testing configuration options.

```
    ALLOWED_CONTENT_TYPES = {'application/json', 'application/octet-stream', 'multipart/fo
```

```
ALLOWED_MIME_TYPES = {'application/pdf', 'application/vnd.ms-excel'}
DATABASE = {'engine': 'peewee.SqliteDatabase', 'name': 'test.db', 'pragmas': {'cach
DEBUG = True
DEVELOPMENT = True
ERROR_404_HELP = False
FLASK_RESTFUL_PREFIX = '/api'
HOME = '/home/docs'
LOGIN_DISABLED = False
LOG_DIRECTORY = '/home/docs/checkouts/readthedocs.org/user_builds/flask-api/checkouts/
MAIL_PASSWORD = None
MAIL_PORT = None
MAIL_SERVER = None
MAIL_USERNAME = None
MAIL_USE_SSL = False
MAIL_USE_TLS = True
RESET_TOKEN_EXPIRES = 86400
RESTX_MASK_SWAGGER = False
ROOT_DIRECTORY = '/home/docs/checkouts/readthedocs.org/user_builds/flask-api/checkouts
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/checko
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_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)
```

CHAPTER

THREE

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.

CHAPTER

FOUR

CHANGELOG

All notable changes to this project will be documented in this file. See standard-version for commit guidelines.

4.1 2.0.0 (2020-10-27)

4.1.1 BREAKING CHANGES

- order field in search requests is a list of dicts.

4.1.2 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)

4.1.3 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)

4.1.4 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)

4.1.5 1.4.1 (2020-10-07)

4.1.6 Bug Fixes

- **celery:** correct problem when start Celery (52ad2fb), closes #3

4.2 1.4.0 (2020-10-04)

4.2.1 Features

- **celery:** add task for exporting several files (ca8355f)
- **documentation:** add sphinx integration (8c313fd)

4.3 1.3.0 (2020-09-20)

4.3.1 Features

- **swagger:** add Swagger full integration (1eaf8d8)

4.4 1.2.0 (2020-09-18)

4.4.1 BREAKING CHANGES

- install/update Node.js and Python libraries

4.4.2 build

- update Node.js and Python packages (b7416cc)

4.5 1.1.0 (2020-05-31)

4.5.1 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)

4.5.2 Refactor

- replace HTTP exceptions to Werkzeug HTTP Exceptions (31e5606)
- move Word and Excel celery tasks to them own modules (00e42e5)

4.5.3 Docs

- docs: add installation project guide (b915d31)

4.6 1.0.0 (2020-05-17)

4.6.1 BREAKING CHANGES

- **pip:** update Python dependencies

4.6.2 Features

- **celery:** add basic installation (147dd2c)
- **db:** add pewee 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)

4.6.3 Build

- **pip:** update requirements.txt (6193153)

4.7 0.8.0 (2020-04-29)

4.7.1 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)

4.8 0.7.0 (2020-04-23)

4.8.1 Features

- **doc:** add standard-version NodeJS package (c1b2cb3)

4.8.2 0.6.1 (2020-04-23)

4.8.3 BREAKING CHANGES

- update python dependencies

4.8.4 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)

4.8.5 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 (aabbc2d)
- **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)

4.8.6 Build

- update requirements.txt (f783e78)
- update requirements.txt (b6378ba)

CHAPTER

FIVE

NOTE

If you find any bugs, odd behavior, or have an idea for a new feature please don't hesitate to on GitHub.

CHAPTER

SIX

INDICES AND TABLES

- genindex
- modindex
- search

PYTHON MODULE INDEX

a

c

d

t

INDEX

Symbols

A

B

C

D

E

F

G

H

I

J

L

M

N

O

P

R

S

T

U

V

W