hobbit-core Documentation

Legolas Bloom

2021 06 21

Contents

changelog // github // pypi // issues // API // EN version

Flask + SQLAlchemy + marshmallow + webargs flask

RESTful APIcelerygitlab-ci/cddocker compose apispec

Convention over configuration

CHAPTER 1

1.1

```
pip install "hobbit-core[hobbit,hobbit_core]"  #
pip install "hobbit-core[hobbit,hobbit_core]" --pre  # pre release
#
pip install "hobbit-core[hobbit]"
```

1.2

hobbit flask:

```
hobbit --echo new -n demo -d . -p 5000 --celery  #  -t rivendell
```

pipenv:

```
pipenv install -r requirements.txt --pre && pipenv install --dev pytest pytest-cov␣
↪pytest-env ipython flake8 ipdb
```

apiserver:

```
pipenv shell  #
FLASK_APP=app/run.py flask run
```

:

```
* Serving Flask app "app/run.py"
* Environment: production
  WARNING: Do not use the development server in a production environment.
  Use a production WSGI server instead.
* Debug mode: off
* Running on http://127.0.0.1:5000/ (Press CTRL+C to quit)
```

http://127.0.0.1:5000/api/ping/

1.3

Hobbit

`hobbit gen` modelscsvmodels.pyCRUD APIunittestrivendell`hobbit create` csv

```
hobbit --echo gen -n user -d /tmp/test -t rivendell --csv-path xx
```

1.4

1. modelsmodelmodel
2. type `sqlalchemy.types` refref `hobbit-core` typestringlength
3. test

`hobbit` :

```
hobbit --help
```

1.5

```
# bash users add this to your .bashrc
eval "$(_HOBBIT_COMPLETE=source hobbit)"
# zsh users add this to your .zshrc
eval "$(_HOBBIT_COMPLETE=source_zsh hobbit)"
```

CHAPTER 2

```
.
├── Dockerfile
├── app
│   ├── __init__.py
│   ├── configs
│   │   ├── __init__.py
│   │   ├── default.py
│   │   ├── development.py
│   │   ├── production.py
│   │   └── testing.py
│   ├── core
│   │   └── __init__.py
│   ├── exts.py
│   ├── models
│   │   └── __init__.py
│   ├── run.py
│   ├── schemas
│   │   └── __init__.py
│   ├── tasks
│   │   └── __init__.py
│   ├── utils
│   │   └── __init__.py
│   └── views
│       ├── __init__.py
│       └── ping.py
├── configs
│   └── gunicorn-logging.ini
├── deploy.sh
├── docker-compose.yml
├── docs
│   └── index.apib
├── pytest.ini
├── requirements.txt
└── tests
    ├── __init__.py
    ├── conftest.py
```

()

()

```
    └── test_ping.py
```

2.1 Dockerfile

dockerwebdocker image Dockerfile reference Dockerfile

2.2 app

app

2.3 configs

flask `FLASK_ENV FLASK_ENV=production configs/production.py`

2.4 core

corehobbit_corehobbit_core

2.5 exts.py

flask `exts.py` Why use exts.py to instance extension?

2.6 models

2.7 services

rivendelljavaviewmodelsevicessevicesmodel

2.8 run.py

web

2.9 schemas

marshmallow scheamsmarshmallowapi `django-rest-framework`

2.10 utils

2.11 views

2.11.1 deploy.sh

ci/cd

2.12 docker-compose.yml

docker compose :

```
docker-compose up
```

2.12.1 docs

API model

2.12.2 logs

log

2.12.3 tests

case. pytest pytest

CHAPTER 3

Others

3.1 Change history

1.4.4 (2020-03-25)

- Fix webargs 6.x.x: limit version < 6.

3.1.1 1.4.3 (2019-07-24)

- Add CustomParser for automatically trim leading/trailing whitespace from argument values(*from hobbit_core.webargs import use_args, use_kwargs*).
- Add *HOBBIT_UPPER_SEQUENCE_NAME* config for upper db's sequence name.
- Fixs some err in template.

3.1.2 1.4.2 (2019-06-13)

- Add *db.BaseModel* for support Oracle id sequence.

3.1.3 1.4.1 (2019-05-23)

- Add template for 4-layers (viewschemaservicemodel).
- Add options api for query all consts defined in *app/models/consts*.
- Add *create* command to generate a csv file that defines some models to use in the *gen* command.
- Removed example code.
- Split hobbit cmd and hobbit_core lib, now install cmd should be *pip install "hobbit-core[hobbit,hobbit_core]"*.
- Remove flask_hobbit when import (*hobbit_core.flask_hobbit.db import transaction –> from hobbit_core.db import transaction*).

- Enhance gen cmd: now can auto create CRUD API and tests.
- Fix typo.
- Update some test cases.

3.1.4 1.4.0 (Obsolete version)

3.1.5 1.3.1 (2019-02-26)

- The strict parameter is removed in marshmallow >= 3.0.0.

3.1.6 1.3.0 (2019-01-14)

- Add import_subs util for auto import modelsschemasviews in module/__init__.py file.
- Add index for created_atupdated_at cloumn and default order_by id.
- Add validate for PageParams.
- Add hobbit gen cmd for auto render views.py, models.py, schemas.py etc when start a feature dev.
- Add ErrHandler.handler_assertion_error.
- Add db.transaction decorator, worked either autocommit True or False.
- pagination return dict instead of class, order_by can set None for
- traceback.print_exc() –> logging.error.
- Foreign key fields support ondelete, onupdate.
- Hobbit startproject cmd support celery option.

3.1.7 1.2.5 (2018-10-30)

- Add ModelSchema(Auto generate load and dump func for EnumField).
- Add logging config file.
- Add EnumExt implementation.
- Fix use_kwargs with fileds.missing=None and enhanced.

3.1.8 1.2.4 (2018-10-18)

- Fix SuccessResult status arg not used.

3.1.9 1.2.3 (2018-10-18)

- Add utils.use_kwargs, fix webargs's bug.

3.1.10 1.2.2 (2018-10-16)

- Add SchemaMixin & ORMSchema use in combination with db.SurrogatePK.
- Now print traceback info when server 500.
- Fix miss hidden files when sdist.

3.1.11 1.2.1 (2018-10-12)

- secure_filename support py2 & py3.

3.1.12 1.2.0 (2018-10-11)

- Gitlab CI/CD support.
- Add secure_filename util.
- Enhance deploy, can deploy to multiple servers.
- Add –port option for startproject cmd.

3.1.13 1.1.0 (2018-09-29)

- Beta release.
- Fix hobbit create in curdir(.) err.
- Add dict2object util.
- Project tree confirmed.
- Add tutorialproject tree doc.
- Add example options for startproject cmd.

3.1.14 1.0.0 (2018-09-25)

- Alpha release.
- flask_hobbit release.

3.1.15 0.0.[1-9]

- hobbit cmd released.
- Incompatible production version.

3.2 Hobbit-core's API Documentation

3.2.1 hobbit cmd

hobbit - A flask project generator.

hobbit.bootstrap.**new**(**args*, ***kwargs*) → Any
Create a new flask project, render from different template.

Examples:

```
hobbit --echo new -n blog -d /tmp/test -p 1024
```

It is recommended to use pipenv to create venv:

```
pipenv install -r requirements.txt && pipenv install --dev pytest pytest-cov␣
↪pytest-env ipython flake8 ipdb
```

hobbit.bootstrap.**gen**(**args*, ***kwargs*) → Any
Generator new feature. Auto gen models/{name}.py, schemas/{name}.py, views/{name}.py, services/{name.py}, tests/test_{name}.py etc.

3.2.2 hobbit_core

A flask extension that take care of base utils.

hobbit_core

Common utils for flask app.

class hobbit_core.**HobbitManager**(*app=None*, *db=None*, ***kwargs*)
Customizable utils management.

init_app(*app*, *db*, ***kwargs*)
app: The Flask application instance.

db

class hobbit_core.db.**BaseModel**(***kwargs*)
Abstract base model class contains `idcreated_at` and `updated_at` columns.

id: A surrogate integer 'primary key' column.

created_at: Auto save `datetime.now()` when row created.

updated_at: Auto save `datetime.now()` when row updated.

Support **oracle id sequence**, default name is `{class_name}_id_seq`, can changed by `sequence_name` and `HOBBIT_UPPER_SEQUENCE_NAME` config. Default value of app.config['HOBBIT_UPPER_SEQUENCE_NAME'] is False.

Examples:

```
from hobbit_core.db import Column, BaseModel

class User(BaseModel):
    username = Column(db.String(32), nullable=False, index=True)

print([i.name for i in User.__table__.columns])
# ['username', 'id', 'created_at', 'updated_at']
```

Can be blocked columns with **exclude_columns**:

```
class User(BaseModel):
    exclude_columns = ['created_at', 'updated_at']
    username = Column(db.String(32), nullable=False, index=True)

print([i.name for i in User.__table__.columns])
# ['username', 'id']
```

Can be changed primary_key's name using **primary_key_name**:

```
class User(BaseModel):
    primary_key_name = 'user_id'
    username = Column(db.String(32), nullable=False, index=True)

print([i.name for i in User.__table__.columns])
# ['username', 'user_id', 'created_at', 'updated_at']
```

Can be changed sequence's name using **sequence_name** (worked with oracle):

```
class User(BaseModel):
    sequence_name = 'changed'
    username = Column(db.String(32), nullable=False, index=True)

# print(User.__table__.columns['id'])
Column('id', ..., default=Sequence('changed_id_seq'))
```

__repr__() → str

You can set label property.

<{classname}({pk}:{label!r})>

str

class hobbit_core.db.**SurrogatePK**

A mixin that add `id`created_at and `updated_at` columns to any declarative-mapped class.

id: A surrogate integer 'primary key' column.

created_at: Auto save `datetime.now()` when row created.

updated_at: Auto save `datetime.now()` when row updated.

It is not recommended. See hobbit_core.db.BaseModel.

__repr__() → str

You can set label property.

<{classname}({pk}:{label!r})>

str

class hobbit_core.db.**EnumExt**

Extension for serialize/deserialize sqlalchemy enum field.

Be sure `type(key)` is `int` and `type(value)` is `str(label = (key, value))`.

Examples:

```
class TaskState(EnumExt):
    # label = (key, value)
    CREATED = (0, '')
    PENDING = (1, '')
    STARTING = (2, '')
```

()

()

```
    RUNNING = (3, '')
    FINISHED = (4, '')
    FAILED = (5, '')
```

classmethod strict_dump(*label: str*, *verbose: bool = False*) → Union[int, str]
Get key or value by label.

Examples:

```
TaskState.strict_dump('CREATED')  # 0
TaskState.strict_dump('CREATED', verbose=True)  # ''
```

Key or value, If label not exist, raise `KeyError`.

int|str

classmethod dump(*label: str*, *verbose: bool = False*) → Dict[str, Any]
Dump one label to option.

Examples:

```
TaskState.dump('CREATED')  # {'key': 0, 'value': ''}
```

Dict of label's key and value. If label not exist, raise `KeyError`.

dict

classmethod load(*val: Union[int, str]*) → str
Get label by key or value. Return val when val is label.

Examples:

```
TaskState.load('FINISHED')  # 'FINISHED'
TaskState.load(4)  # 'FINISHED'
TaskState.load('')  # 'CREATED'
```

Label.

str|None

classmethod to_opts(*verbose: bool = False*) → List[Dict[str, Any]]
Enum to options.

Examples:

```
opts = TaskState.to_opts(verbose=True)
print(opts)

[{'key': 0, 'label': 'CREATED', 'value': u''}, ...]
```

List of dict which key is *key*, *value*, label.

list

class hobbit_core.db.**BaseModelMeta**(*name*, *bases*, *d*)

class hobbit_core.db.**BaseModel**(***kwargs*)

Abstract base model class contains `idcreated_at` and `updated_at` columns.

id: A surrogate integer 'primary key' column.

created_at: Auto save `datetime.now()` when row created.

updated_at: Auto save `datetime.now()` when row updated.

Support **oracle id sequence**, default name is `{class_name}_id_seq`, can changed by `sequence_name` and `HOBBIT_UPPER_SEQUENCE_NAME` config. Default value of app.config['HOBBIT_UPPER_SEQUENCE_NAME'] is False.

Examples:

```
from hobbit_core.db import Column, BaseModel

class User(BaseModel):
    username = Column(db.String(32), nullable=False, index=True)

print([i.name for i in User.__table__.columns])
# ['username', 'id', 'created_at', 'updated_at']
```

Can be blocked columns with **exclude_columns**:

```
class User(BaseModel):
    exclude_columns = ['created_at', 'updated_at']
    username = Column(db.String(32), nullable=False, index=True)

print([i.name for i in User.__table__.columns])
# ['username', 'id']
```

Can be changed primary_key's name using **primary_key_name**:

```
class User(BaseModel):
    primary_key_name = 'user_id'
    username = Column(db.String(32), nullable=False, index=True)

print([i.name for i in User.__table__.columns])
# ['username', 'user_id', 'created_at', 'updated_at']
```

Can be changed sequence's name using **sequence_name** (worked with oracle):

```
class User(BaseModel):
    sequence_name = 'changed'
    username = Column(db.String(32), nullable=False, index=True)

# print(User.__table__.columns['id'])
Column('id', ..., default=Sequence('changed_id_seq'))
```

hobbit_core.db.**reference_col**(*tablename: str*, *nullable: bool = False*, *pk_name: str = 'id'*, *onupdate: str = None*, *ondelete: str = None*, ***kwargs*) → sqlalchemy.sql.schema.Column

Column that adds primary key foreign key reference.

- **tablename** (*str*) – Model.__table_name__.
- **nullable** (*bool*) – Default is False.
- **pk_name** (*str*) – Primary column's name.

- **onupdate** (*str*) – If Set, emit ON UPDATE <value> when issuing DDL for this constraint. Typical values include CASCADE, DELETE and RESTRICT.
- **ondelete** (*str*) – If set, emit ON DELETE <value> when issuing DDL for this constraint. Typical values include CASCADE, DELETE and RESTRICT.

Others:

See `sqlalchemy.Column`.

Examples:

```
from sqlalchemy.orm import relationship

role_id = reference_col('role')
role = relationship('Role', backref='users', cascade='all, delete')
```

hobbit_core.db.**transaction**(*session: sqlalchemy.orm.session.Session*, *nested: bool = False*)

Auto transaction commit or rollback. This worked with `session.autocommit=False` (the default behavior of `flask-sqlalchemy`) or `session.autocommit=True`. See more: http://flask-sqlalchemy.pocoo.org/2.3/api/#sessions

Tips:

- **Can't** do `session.commit()` **in func, otherwise raise** `sqlalchemy.exc.ResourceClosedError`: *This transaction is closed.*
- **Must use the same session in decorator and decorated function**.
- **We can use nested** if keep top decorated by `@transaction(session, nested=False)` and all subs decorated by `@transaction(session, nested=True)`.

Examples:

```
from hobbit_core.db import transaction

from app.exts import db


@bp.route('/users/', methods=['POST'])
@transaction(db.session)
def create(username, password):
    user = User(username=username, password=password)
    db.session.add(user)
    # db.session.commit() error occurred
```

We can nested use this decorator. Must set `nested=True` otherwise raise `ResourceClosedError` (session.autocommit=False) or raise `InvalidRequestError` (session.autocommit=True):

```
@transaction(db.session, nested=True)
def set_role(user, role):
    user.role = role
    # db.session.commit() error occurred


@bp.route('/users/', methods=['POST'])
@transaction(db.session)
def create(username, password):
    user = User(username=username, password=password)
    db.session.add(user)
```

()

()

```
    db.session.flush()
    set_role(user, 'admin')
```

pagination

hobbit_core.pagination.**PageParams = {'order_by': <fields.DelimitedList(default=<marshmallo**

Base params for list view func which contains `pagepage_size order_by` params.

Example:

```
@use_kwargs(PageParams)
def list_users(page, page_size, order_by):
    pass
```

hobbit_core.pagination.**pagination**(*obj: flask_sqlalchemy.model.DefaultMeta, page: int, page_size: int, order_by: Union[str, List[str], None] = 'id', query_exp=None*) → importlib._bootstrap.PaginationType

A pagination for sqlalchemy query.

- **obj** (*db.Model*) – Model class like User.
- **page** (*int*) – Page index.
- **page_size** (*int*) – Row's count per page.
- **order_by** (*str, list, None*) – Example: 'id'['-id', 'column_name'].
- **query_exp** (*flask_sqlalchemy.BaseQuery*) – Query like `User.query.filter_by(id=1)`.

Dict contains `itemspagepage_size` and `total` fileds.

dict

schemas

utils

class hobbit_core.utils.**ParamsDict**

Just available update func.

Example:

```
@use_kwargs(PageParams.update({...}))
def list_users(page, page_size, order_by):
    pass
```

update(*other=None*)

Update self by other Mapping and return self.

class hobbit_core.utils.**dict2object**

Dict to fake object that can use getattr.

Examples:

```
In [2]: obj = dict2object({'a': 2, 'c': 3})

In [3]: obj.a
Out[3]: 2

In [4]: obj.c
Out[4]: 3
```

hobbit_core.utils.**secure_filename**(*filename: str*) → str

Borrowed from werkzeug.utils.secure_filename.

Pass it a filename and it will return a secure version of it. This filename can then safely be stored on a regular file system and passed to `os.path.join()`.

On windows systems the function also makes sure that the file is not named after one of the special device files.

```
>>> secure_filename(u'.zip')
'.zip'
>>> secure_filename('My cool movie.mov')
'My_cool_movie.mov'
>>> secure_filename('../../../etc/passwd')
'etc_passwd'
>>> secure_filename(u'i contain cool ümläuts.txt')
'i_contain_cool_umlauts.txt'
```

hobbit_core.utils.**use_kwargs**(*argmap*, *schema_kwargs: Optional[Dict[KT, VT]] = None*, ***kwargs*)

For fix `Schema(partial=True)` not work when used with `@webargs.flaskparser.use_kwargs`. More details `see webargs.core`.

- **argmap** (*marshmallow.Schema, dict, callable*) – Either a *marshmallow.Schema*, *dict* of argname -> *marshmallow.fields.Field* pairs, or a callable that returns a *marshmallow.Schema* instance.
- **schema_kwargs** (*dict*) – kwargs for argmap.

A dictionary of parsed arguments.

dict

hobbit_core.utils.**import_subs**(*locals_*, *modules_only: bool = False*) → List[str]

Auto import submodules, used in __init__.py.

- **locals** – *locals()*.
- **modules_only** – Only collect modules to __all__.

Examples:

```
# app/models/__init__.py
from hobbit_core.utils import import_subs

__all__ = import_subs(locals())
```

Auto collect Model's subclass, Schema's subclass and instance. Others objects must defined in submodule.__all__.

response

hobbit_core.response.**gen_response**(*code: int*, *message: str = ''*, *detail: Optional[str] = None*) → importlib._bootstrap.RespType

Func for generate response body.

- **code** (*string, int*) – Extension to interact with web pages. Default is http response status_code like 200404.
- **message** (*string*) – For popup windows.
- **detail** (*object*) – For debug, detail server error msg.

A dict contains all args.

dict

class hobbit_core.response.**Result**(*response=None*, *status=None*, *headers=None*, *mimetype='application/json'*, *content_type=None*, *direct_passthrough=False*)

Base json response.

status = 200

class hobbit_core.response.**SuccessResult**(*message: str = ''*, *code: Optional[int] = None*, *detail: Any = None*, *status: Optional[int] = None*)

Success response. Default status is 200, you can cover it by status arg.

status = 200

class hobbit_core.response.**FailedResult**(*message: str = ''*, *code: Optional[int] = None*, *detail: Any = None*)

Failed response. status always 400.

status = 400

class hobbit_core.response.**UnauthorizedResult**(*message: str = ''*, *code: Optional[int] = None*, *detail: Any = None*)

status = 401

class hobbit_core.response.**ForbiddenResult**(*message: str = ''*, *code: Optional[int] = None*, *detail: Any = None*)

status = 403

class hobbit_core.response.**ValidationErrorResult**(*message: str = ''*, *code: Optional[int] = None*, *detail: Any = None*)

status = 422

class hobbit_core.response.**ServerErrorResult**(*message: str = ''*, *code: Optional[int] = None*, *detail: Any = None*)

status = 500

err_handler

class hobbit_core.err_handler.**ErrHandler**

Base error handler that catch all exceptions. Be sure response is:

```
{
    "code": "404",  # error code, default is http status code, you can change it
    "message": "Not found",  # for alert in web page
    "detail": "id number field length must be 18",  # for debug
}
```

Examples:

```
app.register_error_handler(Exception, ErrHandler.handler)
```

h