flask-sqlalchemy-booster Documentation

Release 0.1.0

Surya Sankar

Sep 05, 2018

Contents

1 Features **3**

2 Contents **5**

2.1 Installation 5

2.2 How To Use 5

2.3 Using the Querying API 6

2.4 API 6

3 Indices and tables **7**

Flask-SQLAlchemy-Booster is a collection of enhancements to the Flask-SQLAlchemy library.

It replaces the Model class with a subclass that adds

1. Additional querying methods and
2. Easily configurable `todict` methods and `tojson` methods for serializing objects.

It also provides some decorators and utility functions which can be used to easily generate JSON responses.

CHAPTER 1

Features

- Fully compatible with code written for Flask-SQLAlchemy. It will just transparently replace it with additional features.
- Simple api for most common querying operations:

```
>>> user = User.first()
>>> user2 = User.last()
>>> newcust = Customer.find_or_create(name="Alex", age=21, email="al@h.com")
```

- JSON response functions which can be dynamically configured via the GET request params allowing you to do things like:

```
GET /api/customers?city~=Del&expand=shipments.country,user&sort=desc&limit=5
```

CHAPTER 2

Contents

2.1 Installation

Install via pip:

```
$ pip install Flask-SQLAlchemy-Booster
```

Or you can clone the public repository:

```
$ git clone git@github.com:inkmonk/flask-sqlalchemy-booster.git
```

and then run:

```
$ python setup.py install
```

2.2 How To Use

Since it just subclasses Flask-SQLAlchemy's Model class, the usage is entirely similar.

Set up Flask-SQLAlchemy related configuration keys to set up the database.

Then create a db instance like this:

```
from flask.ext.sqlalchemy_booster import FlaskSQLAlchemyBooster
db = FlaskSQLAlchemyBooster()
```

You can then subclass the `db.Model` class to create your model classes:

```
class User(db.Model):
    id = db.Column(db.Integer, primary_key=True, unique=True)
    email = db.Column(db.String(100), unique=True)
    password = db.Column(db.String(100))
```

(continues on next page)

(continued from previous page)

```
    name = db.Column(db.String(100))
    active = db.Column(db.Boolean())

class Order(db.Model):
    id = db.Column(db.Integer, primary_key=True, unique=True)
user_id = db.Column(db.Integer, db.ForeignKey('user.id'))
```

2.3 Using the Querying API

This module provides a set of commonly used methods - for CRUD operations on models. Usage is very simple

To get the first instance obeying some conditions:

```
customer = Customer.first(state="Rajasthan")

first_cust = Customer.first()
```

To get the last instance obeying some conditions:

```
customer = Customer.last(city="Delhi")

last_cust = Customer.last()
```

2.4 API

2.4.1 Core

2.4.2 ModelBooster

This is the `db.Model` class that you will use as the super class for your Models. It has two sets of methods defined on it, apart from the ones already defined by FlaskSQLAlchemy.

2.4.3 Responses

CHAPTER 3

Indices and tables

- genindex
- modindex
- search