rzhannoy 24 декабря 2013 в 16:35

Peewee – лёгкая, гибкая и очень быстрая ORM на Python

Разработка веб-сайтов*, Python*

Tutorial

> Предлагаю всем джангистам/алхимистам немного отвечься и почитать вольную интерпретацию вводного туториала и частично документации по Peewee – stand-alone ORM, обязательной к ознакомлению любому питонщику и, в особенности, фласкеру. Пишут о ней мало, а зря. С Peewee очень просто подружиться, особенно если вы уже знакомы с какой-нибудь ORM на ActiveRecord. Что более важно – с ней приятно дружить :) Ну, начнём.

Установка

Определение моделей или «попахивает джангой»

Весь нижеследующий код можно повторить один к одному в интерактивном интерпретаторе или отдельном скрипте.

```
from peewee import *

db = SqliteDatabase('people.db')

class Person(Model):
    name = CharField()
    birthday = DateField()
    is_relative = BooleanField()

    class Meta:
        database = db  # модель будет использовать базу данных 'people.db'
```

Типов полей много, на все случаи жизни. Peewee берёт на себя преобразование питоновских объектов в значения, подходящие для базы данных, и наоборот.

Инициализирующие аргументы

Каждое поле принимает следующие инициализирующие аргументы:

- `null=False` – возможно ли хранение null-значений;
- `index=False` – создавать ли индекс для данного столбца в базе;
- `unique=False` – создавать ли уникальный индекс для данного столбца в базе. См. также главу о составных индексах;
- `verbose_name=None` – строка для человекопонятного представления поля;
- `help_text=None` – строка с вспомогательным текстом для поля;
- `db_column=None` – строка, явно задающая название столбца в базе для данного поля, используется например при работе с legacy базой данных;
- `default=None` – значение по-умолчанию для полей класса при инстанцировании;
- `choices=None` – список или кортеж двухэлементных кортежей, где первый элемент – значение для базы, второй – отображаемое значение (аналогично джанге);
- `primary_key=False` – использовать ли данное поле, как первичный ключ;
- `sequence=None` – последовательность для наполнения поля (удостоверьтесь, что бекэнд поддерживает такую функциональность);

Метаданные

Для каждой таблицы можно прописать единые метаданные в `class Meta` :

Опция	Описание	Наследуется?
`database`	база данных для модели	да
`db_table`	название таблицы, в которой будут храниться данные	нет
`indexes`	список полей для индексирования	да
`order_by`	список полей для сортировки по-умолчанию	да
`primary_key`	составной первичный ключ, экземпляр класса CompositeKey, пример	да

`table_alias`	алиас таблицы для использования в запросах	нет

Попробуем задать отношения между моделями через внешний ключ. С peewee это просто:

```
class Pet(Model):
    owner = ForeignKeyField(Person, related_name='pets')
    name = CharField()
    animal_type = CharField()

    class Meta:
        database = db  # модель будет использовать базу данных 'people.db'
```

Модели описаны, осталось создать для них соответствующие таблицы в базе данных:

```
>>> Person.create_table()
>>> Pet.create_table()
```

Работа с данными

Для примера создадим нескольких человек и заведём им домашних животных:

```
>>> from datetime import date
>>> uncle_bob = Person(name='Bob', birthday=date(1960, 1, 15), is_relative=True)
>>> uncle_bob.save()  # сохраним Боба в базе данных
```

Записи можно создавать и напрямую с помощью метода Model.create() без явного save():

```
>>> grandma = Person.create(name='Grandma', birthday=date(1935, 3, 1), is_relative=T
>>> herb = Person.create(name='Herb', birthday=date(1950, 5, 5), is_relative=False)
```

Порадуем бабулю фамилией:

```
>>> grandma.name = 'Grandma L.'
>>> grandma.save()  # обновим запись grandma
```

Теперь сгенерируем немного живности. У бабули аллергия на кошек, а вот у Герба есть некоторые проблемы:

```
>>> bob_kitty = Pet.create(owner=uncle_bob, name='Kitty', animal_type='cat')
>>> herb_fido = Pet.create(owner=herb, name='Fido', animal_type='dog')
>>> herb_mittens = Pet.create(owner=herb, name='Mittens', animal_type='cat')
>>> herb_mittens_jr = Pet.create(owner=herb, name='Mittens Jr', animal_type='cat')
```

В какой-то момент Варежке надоело жить с Гербом и, воспользовавшись открытым окном, он гордо убежал в закат. Уважая его право на свободу личности, всё же удалим соответствующую запись из базы:

```
>>> herb_mittens.delete_instance()  # удачи, Варежка
1
```

Как вы могли заметить, операция удаления возвращает количество удалённых записей, в данном случае – 1.

Дядя Боб решил, что у Герба итак много животных и отжал у него Фидо:

```
>>> herb_fido.owner = uncle_bob
>>> herb_fido.save()
>>> bob_fido = herb_fido  # переименуем переменную для лучшего соответствия суровой
```

Выборки

Выборки выполняются прямо с объектом класса и возвращают экземпляры SelectQuery (аналог QuerySet в джанге).

Извлечение одной записи

Для извлечения одной записи используйте метод `SelectQuery.get()`:

```
>>> grandma = Person.select().where(Person.name == 'Grandma L.').get()
```

Запрос можно сократить, подставив аргумент напрямую в `get()` :

```
>>> grandma = Person.get(Person.name == 'Grandma L.')
```

Извлечение нескольких записей

Пройдемся по всем экземплярам `Person` циклом:

```
>>> for person in Person.select():
...     print person.name, person.is_relative
...
Bob True
Grandma L. True
Herb False
```

Пройдемся по экземплярам `Person` и по всем связанным с ними записями:

```
>>> for person in Person.select():
...     print person.name, person.pets.count(), 'pets'
...     for pet in person.pets:
...         print '    ', pet.name, pet.animal_type
...
Bob 2 pets
    Kitty cat
    Fido dog
Grandma L. 0 pets
Herb 1 pets
    Mittens Jr cat
```

Выловим всех кошек и их хозяев людей (или наоборот?):

```
>>> for pet in Pet.select().where(Pet.animal_type == 'cat'):
...     print pet.name, pet.owner.name
...
```

```
Kitty Bob
Mittens Jr Herb
```

Не без join'ов:

```
# выберем всех животных Боба
>>> for pet in Pet.select().join(Person).where(Person.name == 'Bob'):
...     print pet.name
...
Kitty
Fido
```

Извлечь ту же выборку можно и по-другому – явно передав объект с Бобом в запрос:

```
>>> for pet in Pet.select().where(Pet.owner == uncle_bob):
...     print pet.name
```

Упорядочим выборку в алфавитном порядке. Для этого воспользуемся методом `SelectQuery.order_by()` :

```
>>> for pet in Pet.select().where(Pet.owner == uncle_bob).order_by(Pet.name):
...     print pet.name
...
Fido
Kitty
```

Упорядочим людей по возрасту:

```
>>> for person in Person.select().order_by(Person.birthday.desc()):
...     print person.name
...
Bob
Herb
Grandma L.
```

Давайте попробуем более сложный запрос. Выберем всех людей, родившихся

- до 1940
- после 1959

```
>>> d1940 = date(1940, 1, 1)
>>> d1960 = date(1960, 1, 1)
>>> for person in Person.select().where((Person.birthday < d1940) | (Person.birthday
...     print person.name
...
Bob
Grandma L.
```

Хинт

А теперь торобоан. Выберем тех, кто родился между 1940 и 1960:

```
>>> for person in Person.select().where((Person.birthday > d1940) & (Person.birthday
...     print person.name
...
Herb
```

And one last thing. Воспользуемся SQL-функцией и выберем всех людей, чьё имя начинается с «G» в любом регистре:

```
>>> for person in Person.select().where(fn.Lower(fn.Substr(Person.name, 1, 1)) == 'g
...     print person.name
...
Grandma L.
```

Для выборок также используйте методы:

- `SelectQuery.group_by()`
- `SelectQuery.having()`

- `SelectQuery.limit()` и `SelectQuery.offset()`

Если вам понравился данный краткий туториал, обязательно посетите официальную документацию — там много интересного, включая рецепты с решениями распространённых задач и набор плагинов, расширяющих базовую функциональность.

Бонус

Автора в его блоге спросили о быстродействии ORM, на что тот ответил:

> On my machine peewee has been faster than Django and SQA at most tasks, and about the same when iterating and returning Model instances.

> На моём компе peewee обошла Django и SQLAlchemy на большинстве задач, и показала сравнимые результаты на итерациях и выборке инстансов.

После чего опубликовал результаты бенчмарка на гитхабе. Тестились обычные модели и связанные через ForeignKey в различных сценариях. Весьма любопытно.

Кому интересно, исходники:

- Django: модели и бенчмарк
- SQLAlchemy: модели и бенчмарк
- Peewee: модели и бенчмарк
- Сводный бенч

Хорошая альтернатива Алхимии, как считаете?

Теги: python, питон, peewee, orm, flask, бенчмарк, базы данных

Хабы: Разработка веб-сайтов, Python

24 Карма 0 Рейтинг

Николай @rzhannoy

Веб-фулстак

Комментарии 46

Публикации

ЛУЧШИЕ ЗА СУТКИ ПОХОЖИЕ

alexopryshko сегодня в 17:58

Школу закончил в 14, Бауманку в 18: почему, зачем и какие последствия

+43 8.3K 21 61 +61

MaFrance351 сегодня в 15:01

Считываем и эмулируем карты с магнитной полосой

+41 3.4K 29 19 +19

elena_pastukhova сегодня в 19:01

Как мы съедаем 15 тонн воды в день

+29 3.7K 7 12 +12

Sagidullin сегодня в 03:53

Big bada boom отменяется? Подводные интернет-магистрали выдержат наступление «События Кэррингтона»

+29 2.5K 5 12 +12

sergey__pushkin сегодня в 14:01

Язык диаграмм

+26 2.3K 26 7 +7

Просодия в синтезе речи: как мы заставили Ленина спеть «Крылатые качели»

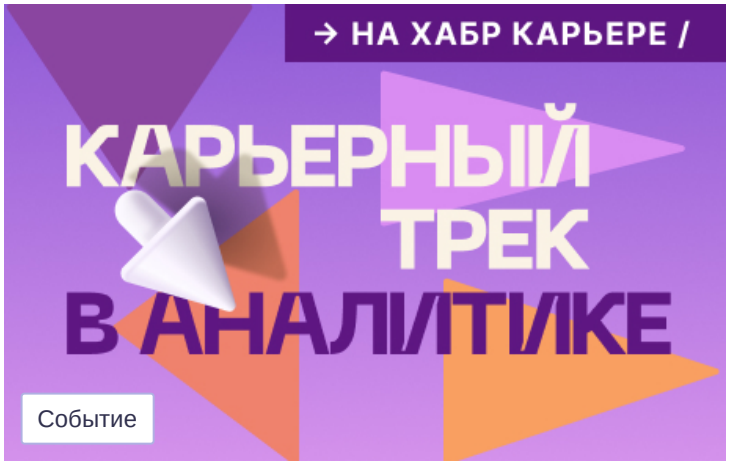
Турбо

МИНУТОЧКУ ВНИМАНИЯ

Разместить

Мегатест

Быстрый и производительный: iOS против Android



Событие

Постройте карьерный трек в аналитике

ВАКАНСИИ

Senior Backend Developer
от 250 000 до 380 000 ₽ · АйТи Бастион · Можно удаленно

Python-разработчик (направление B2B2C)
от 200 000 до 400 000 ₽ · SberDevices · Москва

Senior Software Engineer (Python)
от 5 000 до 6 000 € · Exness · Лимассол · Можно удаленно

Программист PHP 7.3+
от 60 000 до 100 000 ₽ · Ростелеком · Можно удаленно

DBA PostgreSQL (Администратор баз данных PostgreSQL)
до 300 000 ₽ · SberTech · Новосибирск

Больше вакансий на Хабр Карьере

ЧИТАЮТ СЕЙЧАС

Как школьники МЭШ взломали

34K 69 +69

Школу закончил в 14, Бауманку в 18: почему, зачем и какие последствия

8.3K 61 +61

Ловушка алгоритмизации, или как 44-ФЗ породил коррупцию

2.6K 38 +38

Lumia 640 — всё ещё достоин?

2K 11 +11

Как мы съедаем 15 тонн воды в день

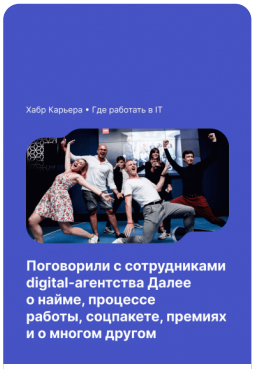
3.7K 12 +12

До конца года можно выбить ачивку Технотекста 2022

Событие

ИСТОРИИ

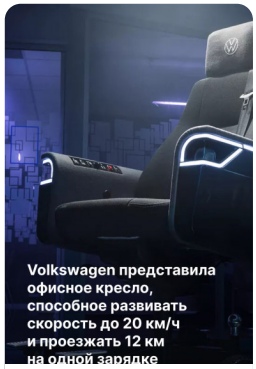
Причины возгорания аккумулятора



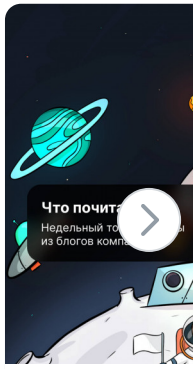
Где работать в IT: Далее

Итоги сезона Data Mining

Не работай через силу



Офисное кресло Volkswagen



Недельный то годноты от компаний

РАБОТА

Python разработчик
164 вакансии

Django разработчик
49 вакансий

Data Scientist
122 вакансии

Все вакансии