


بسم الله الرحمن الرحيم

# مهران کامرانی

روش های بکاپ گرفتن

Mehran

# ساختار

	mehran1421 add readme.md	99301
folder	backup	change save directory to another place
folder	cron	create cron app for schedule for backup per day
folder	dbBackup	testing
folder	dumpData	work with dumpdata and create db.json and db.yaml
folder	media/images	work with dumpdata and create db.json and db.yaml
file	.gitignore	initial commit
file	Readme.md	add readme.md
file	celerybeat-schedule.bak	create cron app for schedule for backup per day
file	celerybeat-schedule.dat	change save directory to another place
file	celerybeat-schedule.dir	create cron app for schedule for backup per day
file	db.json	work with dumpdata and create db.json and db.yaml
file	db.sqlite3	create cron app for schedule for backup per day
file	db.yaml	work with dumpdata and create db.json and db.yaml
file	manage.py	initial commit
file	requirements.txt	add readme.md

از دو روش برای بکاپ گیری استفاده شده که برای هر روش app و models.py های مختلف ساخته شده

# \*\*\*\*\*Django-dbbbackup روش\*\*\*\*\*

dbBackup برای روش Django-dbbbackup ساخته شده و شامل موارد زیر است:

```
class Category(models.Model):
    title = models.CharField(max_length=200)
    slug = models.SlugField(max_length=100)
    status = models.BooleanField(default=True)
    position = models.IntegerField()

    class Meta:
        ordering = ['position']

    def __str__(self):
        return self.title

class Post(models.Model):
    title=models.CharField(max_length=200)
    thumbnail=models.ImageField(upload_to='images')
    category = models.ManyToManyField(Category, null=True)
    created = models.DateTimeField(auto_now_add=True)

    def __str__(self):
        return self.title

class Test(models.Model):
    name=models.CharField(max_length=200)

    def __str__(self):
        return self.name
```

دو جدول ساختیم که برای بکاپ گیری استفاده کنیم

در این روش ما از سلری برای اینکه به طور اتوماتیک از دیتاهای موجود بکاپ بگیرد استفاده میکنیم

در پوشه اصلی پروژه تنظیمات مربوط به سلری را مشاهده میکنیم

```
1 import os
2
3 from celery import Celery
4
5 # set the default Django settings module for the 'celery' program.
6 os.environ.setdefault('DJANGO_SETTINGS_MODULE', 'backup.settings')
7
8 app = Celery('backup')
9
10 # Using a string here means the worker doesn't have to serialize
11 # the configuration object to child processes.
12 # - namespace='CELERY' means all celery-related configuration keys
13 #   should have a `CELERY_` prefix.
14 app.config_from_object('django.conf:settings', namespace='CELERY')
15
16 app.conf.beat_schedule={
17     'get_backup_data_30s':{
18         'task':'cron.tasks.backup',
19         'schedule':10
20     }
21 }
22
23 # Load task modules from all registered Django app configs.
24 app.autodiscover_tasks()
```

که در اینجا هر 10 ثانیه هست که میتوانیم عدد را به دلخواه تعیین کنیم

```
CELERY_BROKER_URL = 'redis://localhost:6379'
CELERY_RESULT_BACKEND = 'redis://localhost:6379'
```

برای بروکر از ردیس استفاده میکنیم که با داکر ایمیج آن را بالا آوردیم

اگر نخواهیم از عکس بکاپ بگیریم روش `dumpdata` بسیار بهتراست ولی اگر فایل های مدیا نیز در پروژه مهم باشند باید از پکیج زیر استفاده کنیم برای بکاپ گیری باید پکیج مربوطه را نصب کنیم:

```
pip install django-dbbackup
```

کانفیگ های لازم را انجام میدم و تو قسمت ادمین یک آبجکت اضافه میکنیم

✔ The post "واکسن کرونا، دربی را تحت تاثیر قرار می دهد؟" was added successfully.

Select post to change

Action:  Go 0 of 1 selected

☐ POST

☐ واکسن کرونا، دربی را تحت تاثیر قرار می دهد؟

1 post

در ترمینال دستور زیر را اجرا میکنیم تا از این آبجکت ذخیره شده یک بکاپ بگیرد

```
$ python manage.py dbbackup
```

یک آبجکت دیگر را اضافه میکنیم:

✔ The post "عجیب اما واقعی؛ سرمربی بارسلونا تحت هر شرایطی اخراج است" was added successfully.

Select post to change

Action:  Go 0 of 2 selected

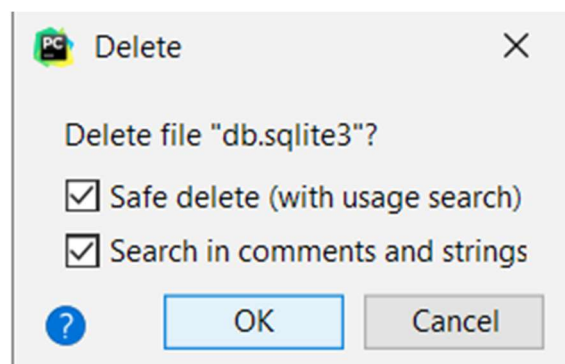
☐ POST

☐ عجیب اما واقعی؛ سرمربی بارسلونا تحت هر شرایطی اخراج است

☐ واکسن کرونا، دربی را تحت تاثیر قرار می دهد؟

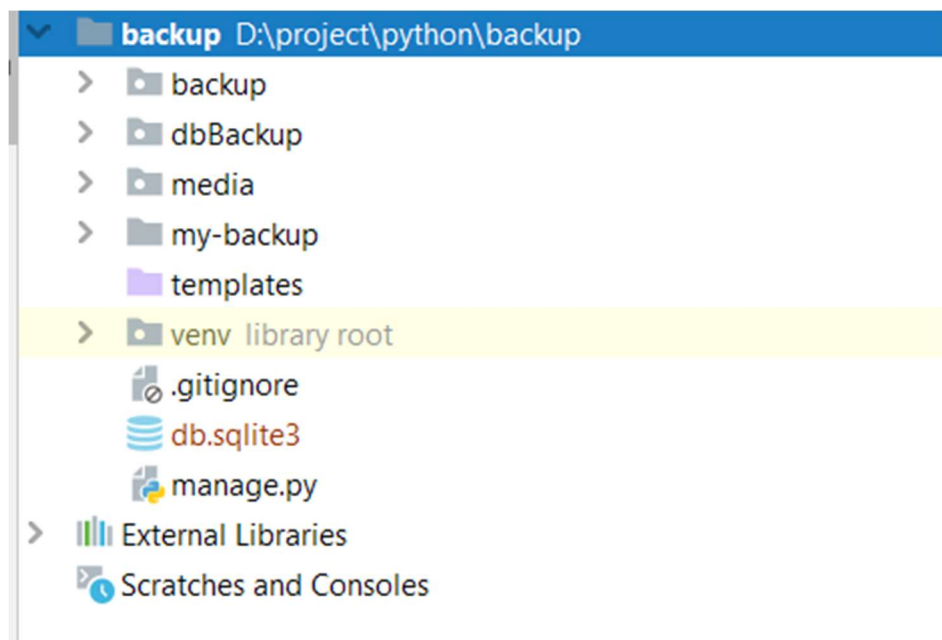
2 posts

حالا دیتابیس رو حذف میکنیم



با دستور زیر میتوانیم دیتابیس را برگردانیم با همان اطلاعات قبلی:

```
(venv) D:\project\python\backup>python manage.py dbrestore
System check identified some issues:
```



علامت قرمز رنگ نشان دهنده ی ایجاد فایل جدید است

برنامه را اجرا میکنیم ،اطلاعات آن با وجود حذف دیتابیس باقی مانده، همان اطلاعاتی که قبل از اجرای دستور بکاپ بود یعنی ابجکت اول ولی ابجکت دوم چون دستور بکاپ را وارد نکردیم حذف شد

---

Select post to change

Action:   0 of 1 selected

☐ POST

☐ واکسن کرونا، دربی را تحت تاثیر قرار می دهد؟

---

1 post

---

و اطلاعات مثل عکس در این روش به راحتی هندل میشود

خب ما می‌خواهیم به طور اتوماتیک این بکاپ اجرا شود بنابراین باید تسک زیر را بنویسیم:

```
1 from celery import shared_task
2 import os
3 from django.core import management
4
5
6 @shared_task
7 def backup():
8     management.call_command('dbbackup')
```

و وقتی با سلری برنامه را اجرا می‌کنیم هر 10 ثانیه این اتفاق انجام می‌شود

```
pip3 install -r requirements.txt
python manage.py makemigrations
python manage.py migrate
python manage.py runserver
celery -A backup beat -l info
celery -A backup worker -P gevent --loglevel=INFO
```

برای بکاپ گرفتن از فایل‌های media باید دستور زیر را اجرا کنیم:

```
$ ./manage.py mediabackup
Backup size: 10.0 KiB
Writing file to zuluvm-2016-07-04-081612.tar
```

حالا با حذف فایل‌های عکس دیگر نمیتوانیم به آن فایل‌ها دسترسی داشته باشیم مگر اینکه دستور زیر را بزنیم



```
$ ./manage.py mediarestore
Restoring backup for media files
Finding latest backup
Reading file zulum-2016-07-04-082551.tar
Restoring: zulum-2016-07-04-082551.tar
Backup size: 10.0 KiB
Are you sure you want to continue? [Y/n]
2 file(s) restored
```

حالا میتوانیم عکس های حذف شده را ببینیم در پنل ادمین:



عکس بالا بعد از restor کردن media است

برای بکاپ نگرفتن از یک جدول خاص باید این کد را به settings.py اضافه کنیم

```

DBBACKUP_CONNECTORS = {
    'default': {
        'USER': 'mehran',
        'PASSWORD': '',
        'HOST': '',
        'EXCLUDE': 'dbBackup_test'
    }
}

```

و حالا dbbackup را انجام می‌دهیم و از این جدول هیچ بکاپی نمی‌گیرد  
 برای اینکه هر چند روز یک بار بکاپ‌های قدیمی را حذف کند باید این خط را اضافه  
 کنیم:

```

DBBACKUP_CONNECTORS = {
    'default': {
        ...
        'CLEANUP_KEEP': 5 #1,2,3,... day
    }
}

```

که می‌گوید هر فایل بکاپ را فقط 5 روز نگهداری کنه  
 برای اینکه محل ذخیره سازی در پوشه اصلی نباشد :

```

DBBACKUP_STORAGE = 'django.core.files.storage.FileSystemStorage'
DBBACKUP_STORAGE_OPTIONS = {'location': 'd://my-backup'}

```

در قسمت value در دیکشنری باید مکان ذخیره شدن رو بنویسیم

---

## روش dumpdata

برای روش dumpdata یک app جداگانه نصب میکنیم

```
Python manage.py dumpdata --indent 2 dumpData.Post > db.json
```

اطلاعات مربوط به این جدول رو به صورت یک فایل json ذخیره میکند ولی یه مشکلی داره اینکه که اطلاعات اگه فارسی باشه نمیتونه به json تبدیل کنه

برای exclude کردن :

```
(venv) D:\project\python\backup>python manage.py dumpdata --indent 2 --exclude dumpData.Test dumpData > db.json
```

---

برای ذخیره کردن در یک فایل yaml :

```
(venv) D:\project\python\backup>python manage.py dumpdata --indent 2 --format=yaml --exclude dumpData.Test dumpData > db.yaml
```

که قبلش بهتر است pyyaml را نصب کنیم

```
- model: dumpData.category
  pk: 1
  fields:
    title: addd
    slug: adel
    status: true
    position: 3
- model: dumpData.post
  pk: 1
  fields:
    title: mehrandldikfjwjijdfolikjolwi
    thumbnail: images/1_WCgsBB6.jpg
    category:
      - 1
```

با دستور

Python manage.py flush

دسیتابیس رو حذف میکنیم و میخوایم اطلاعاتمان رو از فایل db.json که ساخته بودیم لود کنیم

Python manage.py loaddata db.json

مشکل این روش این است که نمیتواند فایل های مدیا را بکاپ بگیرد

---

## بررسی فایل های ایجاد شده

celerybeat-schedule.bak	create cron app for schedule for backup per day
celerybeat-schedule.dat	change save directory to another place
celerybeat-schedule.dir	create cron app for schedule for backup per day

این فایل بعد از اجرای celery beat ایجاد شده شامل کانفیگ ها و اطلاعاتی است که خود celery آن را به وجود می آورد و به صورت داینامیک تولید شده است

db.json	work with dumpdata and create db.json and db.yaml
db.sqlite3	create cron app for schedule for backup per day
db.yaml	work with dumpdata and create db.json and db.yaml

## فایل db.json بعد از اجرای دستور

```
python manage.py dumpdata --indent 2 dumpData.Post > db.json
```

```
1  [
2  {
3      "model": "dumpData.category",
4      "pk": 1,
5      "fields": {
6          "title": "addd",
7          "slug": "adel",
8          "status": true,
9          "position": 3
10     }
11 },
12 {
13     "model": "dumpData.post",
14     "pk": 1,
15     "fields": {
16         "title": "mehrاندldikfjwjijdfolikjolwi",
17         "thumbnail": "images/1_WCgsBB6.jpg",
18         "category": [
19             1
20         ]
21     }
22 }
23 ]
```

ایجاد شده است و شامل اطلاعات مربوط به جدول هایی است که در کد بالا نوشتیم که بکاب بگیرد

## فایل db.yaml

شامل همان اطلاعات ولی به صورت فرمت yaml که بعد از دستور زیر ایجاد شده است

```
python manage.py dumpdata --indent 2 dumbData.Post > db.yaml
```

```
1  - model: dumpData.category
2    pk: 1
3    fields:
4      title: addd
5      slug: adel
6      status: true
7      position: 3
8  - model: dumpData.post
9    pk: 1
10   fields:
11     title: mehrandldikfjwjijdfolikjolwi
12     thumbnail: images/1_WCgsBB6.jpg
13     category:
14       - 1
```