

## 任务调度框架APScheduler

安装：  
(proj\_b) [root@backup-platform sched]# pip install apscheduler

一．四个主要组件：

1. trigger：触发器定义任务以何种条件触发
2. jobstore：任务存储的仓库，持久化任务存储
3. executor：任务执行器它是执行任务的模块，不同的IO模型选择相适应的executor
4. scheduler:任务控制器

七种scheduler：

BlockingScheduler、BackgroundScheduler、AsyncIOScheduler、GeventScheduler、TornadoScheduler、TwistedScheduler、QtScheduler

四种jobstore：  
MemoryJobStore、sqlalchemy、mongodb、redis

三种trigger：

1. date：固定的日期触发器：任务只运行一次，运行完毕后自动清除，若错过指定运行的时间，任务不会创建
2. interval：时间间隔触发器 (间隔多长时间执行一次任务)
3. cron：crontab风格的任务触发

简单应用：

创建一个调度任务的方法1：

```
In [ ]: import time
from apscheduler.schedulers.blocking import BlockingScheduler

def my_job():
    print(time.time())

sched = BlockingScheduler()
sched.add_job(my_job, 'interval', seconds=2, args=[])
sched.start()
```

以上代码每隔5s执行一次my\_job函数，输出当前时间信息。

方法2：

```
In [ ]: import time
from apscheduler.schedulers.blocking import BlockingScheduler

sched = BlockingScheduler()

@sched.scheduled_job('interval', seconds=2)
def my_job():
    print(time.time())

sched.start()
```

获取job列表

获得调度作业的列表可以使用get\_jobs()完成，它会返回所有job的实例。在get\_jobs()中传入任务ID可以获取指定任务的作业列表。

```
In [ ]: import time
from apscheduler.schedulers.blocking import BlockingScheduler

def my_job():
    print(time.time())

sched = BlockingScheduler()
job = sched.add_job(my_job, 'interval', seconds=2 ,id='123')
print(sched.get_job(job_id='123'))
print(sched.get_jobs())
```

移除作业

```
In [ ]: import time
from apscheduler.schedulers.blocking import BlockingScheduler

sched = BlockingScheduler()
def my_job():
    print(time.time())

job = sched.add_job(my_job, 'interval', seconds=2 ,id='123')
sched.add_job(my_job, 'interval', minutes=2, id='my_job_id')
print(sched.get_jobs())
sched.remove_job('my_job_id')
print(sched.get_jobs())
job.remove()
print(sched.get_jobs())
```

```
In [ ]: 关闭调度器：
sched.shutdown()
sched.shutdown(wait=False)
```

默认情况下调度器会等待所有正在运行的作业完成后，再关闭所有的调度器和作业存储，如果不想等待可以将wait选项设置为False。

作业运行的控制(trigger)

add\_job的第二个参数是Trigger，它管理着作业调度的方式。可以为date，interval或cron，对于不同的trigger对应的参数也是不同的。

1. Cron所需要的时间参数：

(int|str) 表示参数既可以是int类型，也可以是str类型  
(datetime | str) 表示参数既可以是datetime类型，也可以是str类型

year (int|str) - 4-digit year - (表示四位数的年份，如2008年)  
month (int|str) - month (1-12) - (表示取值范围为1-12月)  
day (int|str) - day of the (1-31) - (表示取值范围为1-31日)  
week (int|str) - ISO week (1-53) - (格里历2006年12月31日可以写成2006年-w52-7 (扩展形式) 或2006w527 (紧凑形式))  
day\_of\_week (int|str) - number or name of weekday (0-6 or mon,tue,wed,thu,fri,sat,sun) - (表示一周中的第几天，既可以用0-6表示也可以用其英语缩写表示)  
hour (int|str) - hour (0-23) - (表示取值范围为0-23时)  
minute (int|str) - minute (0-59) - (表示取值范围为0-59分)  
second (int|str) - second (0-59) - (表示取值范围为0-59秒)  
start\_date (datetime|str) - earliest possible date/time to trigger on (inclusive) - (表示开始时间)  
end\_date (datetime|str) - latest possible date/time to trigger on (inclusive) - (表示结束时间)

timezone (datetime.tzinfo|str) – time zone to use for the date/time calculations (defaults to scheduler timezone) – (表示时区取值)

示例:

```
#表示2017年3月22日17时19分07秒执行该程序
sched.add_job(my_job, 'cron', year=2017,month = 03,day = 22,hour = 17,minute = 19,second = 07)
```

```
#表示任务在6,7,8,11,12月份的第三个星期五的00:00,01:00,02:00,03:00 执行该程序
sched.add_job(my_job, 'cron', month='6-8,11-12', day='3rd fri', hour='0-3')
```

```
#表示从星期一到星期五5:30 (AM) 直到2014-05-30 00:00:00
sched.add_job(my_job(), 'cron', day_of_week='mon-fri', hour=5, minute=30,end_date='2014-05-30')
```

```
#表示每5秒执行该程序一次, 相当于interval 间隔调度中seconds = 5
sched.add_job(my_job, 'cron',second = '*/5')
```

2. Interval间隔调度

2.1 所需要的时间参数

weeks (int) – number of weeks to wait  
days (int) – number of days to wait  
hours (int) – number of hours to wait  
minutes (int) – number of minutes to wait  
seconds (int) – number of seconds to wait  
start\_date (datetime|str) – starting point for the interval calculation  
end\_date (datetime|str) – latest possible date/time to trigger on  
timezone (datetime.tzinfo|str) – time zone to use for the date/time calculations

示例:

```
表示每隔3天17时19分07秒执行一次任务
sched.add_job(my_job, 'interval',days = 03,hours = 17,minutes = 19,seconds = 07)
```

3. date定时调度(作业只会执行一次)

示例:

```
sched.add_job(my_job, 'date', run_date=date(2009, 11, 6), args=['text'])
```