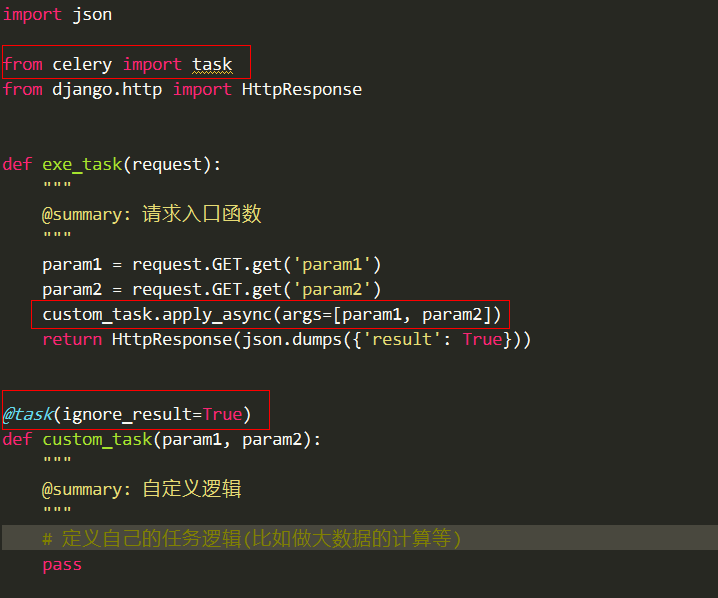
# django+celery的使用介绍

celery是一个异步任务队列/基于分布式消息传递的作业队列，它可以支持很多任务类型，例如普通任务，周期性任务，定时任务等。有兴趣的读者可以参阅[官方文档](http://docs.celeryproject.org/en/latest/index.html)，本文档只介绍常用的几类使用方法。*注意：celery的相关配置，请参考“蓝鲸SaaS开发框架”中的settings部分。*

## 一、普通后台任务

后台任务通过将同步请求异步化，可以有效地解决请求超时的问题（比如SaaS中有一个执行任务的请求，该任务执行耗时2分钟）。

* **普通后台任务代码示例**：



如上图的代码片段，调用custom\_task.apply\_async(args=[param1, param2])时，不会立即执行custom\_task函数，而是向消息队列中插入一条相关的任务元数据，接着程序会立即执行return操作。同时，celery服务端会从消息队列中取出这条元数据并执行custom\_task。这样就实现了耗时任务的异步化，空闲出了uwsgi的worker资源。

* **本地开发时celery的启动方法**：

python manage.py celery worker --settings=settings

## 二、周期性任务

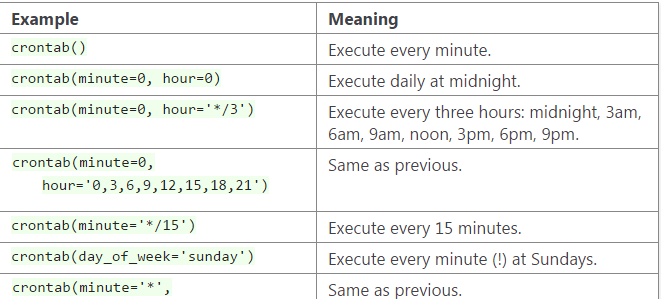
周期性任务是较为常见的后台任务，例如周期地推送消息，周期地清理日志等。 celery的周期性任务用法类似于linux的crontab任务。由于它是应用层面的，可用性更灵活，用户可以通过简单的配置数据库，就可以管理周期性任务了。

* **示例**：定期清理日志文件。



如上图中的代码片段，clean\_log1会在celery beat启动后，自动注册到数据库中(CELERYBEAT\_SCHEDULER = "djcelery.schedulers.DatabaseScheduler")。clean\_log2则需要手动配置到库中(开发框架中提供了api接口来直接操作数据库)。区别在于两个装饰器的作用不同，@task是通用的，而@periodic\_task是针对周期性任务的，参数run\_every设置了周期crontab。

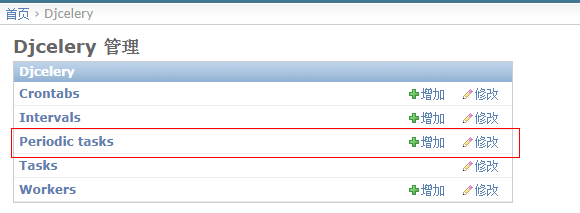
crontab设置规则类似于linux系统的crontab。具体的配置可以[参考文档](http://celery.readthedocs.org/en/latest/userguide/periodic-tasks.html)。

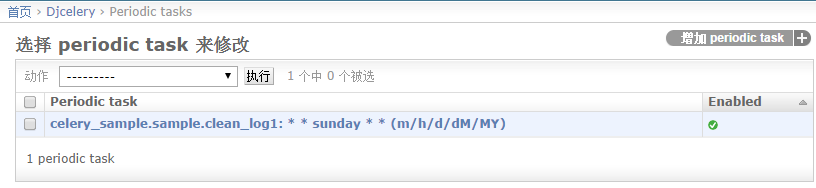


* **配置出周期性任务：**

celery执行周期性任务的一大优势是可配置性，即不修改原有代码，就可以方便的调整任务的调度规则和关闭任务等。和app的自定义model管理方式一样，Djcelery库也可以通过admin页面管理，如下图，它是Djcelery的库，其中Crontabs和Interval是配置调度规则的，Periodic tasks是配置任务的。强调一点，这里的任务都需要在代码中用装饰器@task或者@periodic\_task包装，否则是无效的。

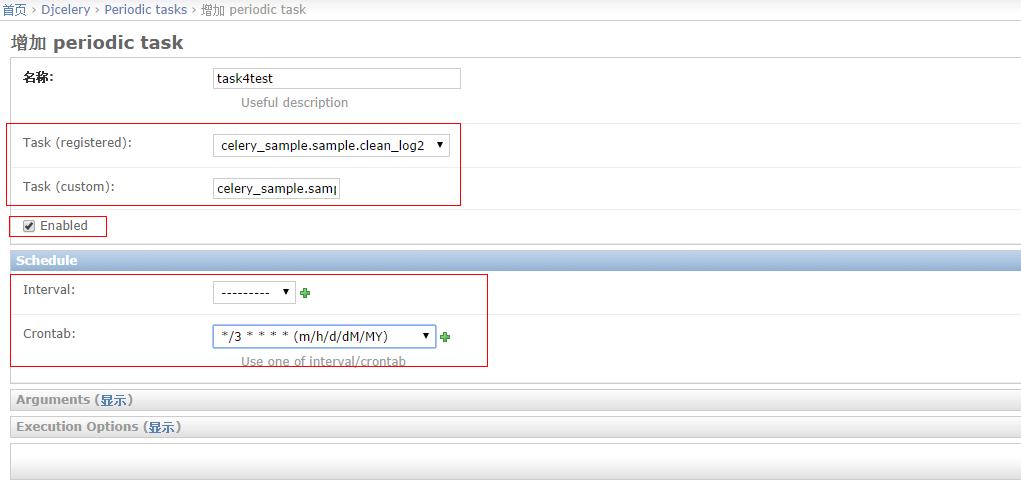
点击Periodic tasks后， celery beat启动后，自动注册了clean\_log1任务。如果需要将clean\_log2配置成周期性任务，可以通过配置Crontabs





和Periodic tasks实现。同时，平台也提供了对应的api方便开发者使用。

下图是一个配置示例，其中Task是注册上clean\_log2, Crontab选择的是每3分钟执行一次。



* **本地开发时celery周期性任务的启动方法**：

python manage.py celery worker --settings=settings

python manage.py celerybeat --settings=settings

## 三、定时任务

指定定时的时间点，执行某个任务。

* **示例**：定时发送邮件。



上图中给出了调用定时任务的两种例子。

countdown=60和eta=datetime.now()+timedelta(seconds=60)都是设置send\_email\_task在60s后执行。

**Celery的更多用法可参考官方文档。**