celery的消息队列

异步任务（Async Task）在业务逻辑中被触发发往消息中间件,定时任务（Celery Beat）周期性的去执行发往消息中间件（Broker）,任务单元实时监控消息中间件,任务执行单元将任务执行结果存储到 Backend 中。

消息中间件和任务执行结果可以用不同的存储方式,

版本

```
celery==3.1.26
celery-with-redis==3.0
django==2.1.5
django-celery==3.2.2
kombu==3.0.37
redis=3.2.12
```

先在子应用下写个任务(以发邮件为例)

task.py 中

```
import time
from django.core.mail import send_mail
from celery import task
from ok import settings


@task
def send(email):
 time.sleep(10) # 测试使用
  token = 'ok'
  title = '村口集合'
  content = '<a href="http://127.0.0.1:8000/user/active/?token=' + token + '">激活账号</a>'
  send_mail(title, content, settings.EMAIL_FORM, [email], html_message=content)
  return True
```

切换到redis的安装目录,运行

```
redis-server ./redis.windows.conf
```

项目的 settings.py中

在这里指定消息队列和结果的url

```
INSTALLED_APPS = [
 'django.contrib.admin',
 'django.contrib.auth',
 'django.contrib.contenttypes',
 'django.contrib.sessions',
 'django.contrib.messages',
 'django.contrib.staticfiles',
 'admin01.apps.Admin01Config',
 'rest_framework',
  'djcelery', # 注册子应用
 ]
 ......

 import djcelery
 djcelery.setup_loader()
 BROKER_URL = 'redis://127.0.0.1:6379/1' # 消息队列
 CELERY_IMPORTS = ('app01.task') # 任务路径
 CELERY_RESULT_BACKEND = 'redis://127.0.0.1:6379/2' # 结果存储
```

执行数据库迁移;

```
python manage.py migrate
```

在视图函数内调用,要给Worker发送任务，需要调用 delay() 方法;

```
class SendMailAPIView(APIView):
 def get(self, request):
 ret = {}
 send.delay('2211217830@qq.com')
 ret['code'] = 200
 ret['message'] = '成功'
 return Response(ret)
```

在manage.py中添加一下代码

```
import django

if __name__ == '__main__':
 os.environ.setdefault('DJANGO_SETTINGS_MODULE', 'ok.settings')
 django.setup()
```

```
    pass

```

启动 worker

```
python manage.py celery worker --loglevel=info
```

处理报错

错误一:如果报错找不到 "async"包

1. 先到 "Python\Python37\site-packages\kombu"目录下更改"async"文件夹名称,改成其他名称即可;
2. 重启worker,修改代码 :将所有"async"改为,上一步新命的名;直至不再报与"async"相关的错误.

错误二:'Command' has no attribute 'option_list'

1. 修改代码: 在 "Command"类下添加属性 option_list=()

```
class Command(models.Model)):
    option_list=() # 给这个类加个属性
    pass
```

错误三：worker 可以正常启动，但是接不到任务，

具体信息：celery AttributeError: 'str' object has no attribute 'items'

Redis 版本过高 ,降低其版本

警告一:Using settings.DEBUG leads to a memory leak ,使用DEBUG调试会导致内存泄漏,

在settings.py 中:

```
DEBUG = False
ALLOWED_HOSTS = ['*']
```

重启项目

```
python manage.py runserver
```

小结

Redis

> 可能因意外中断或者电源故障导致数据丢失的情况

在docker上拉取Redis镜像

拉取

```
docker pull redis:3.2.12
```

运行

```
docker run -p 6379:6379 -v $PWD/data:/data -d redis:3.2 redis-server --ap
pendonly yes
```

在"安全组设置中"暴露出端口号

在settings.py中指定 消息队列的路径,和结果路径

```
BROKER_URL = 'redis://116.62.155.103/1' # 消息队列
CELERY_RESULT_BACKEND = 'redis://116.62.155.103/2' # 结果存储
```

RabbitMQ

> 功能完备，稳定的并且易于安装的broker. 它是生产环境中最优的选择

在docker上拉取RabbitMQ镜像

拉取

```
docker pull rabbitmq # 创建新容器
```

运行

```
docker run -d -p 5672:5672 --name myrabbit rabbitmq
```

- -d 在run后面加上-d参数,则会创建一个守护式容器在后台运行(这样创建容器后不会自动登录容器，如果只加-i -t 两个参数，创建后就会自动进去容器)。
- -p 表示端口映射，前者是宿主机端口，后者是容器内的映射端口。可以使用多个-p 做多个端口映射

可用 docker ps 查看刚才创的容器,创建成功后 创建用户,添加虚拟环境并分配权限

进入容器

```
docker container exec -it myrabbit /bin/bash
```

新增用户,rabbitmqctl list_users获取用户列表

```
rabbitmqctl add_user username01 password01
# 这个用户名,密码在settings.py中使用
```

设置管理员;

```
rabbitmqctl set_user_tags username01 administrator
```

添加虚拟环境,使用"rabbitmqctl list_vhosts"获取添加过的虚拟环境;

```
rabbitmqctl add_vhost myvhost
# myvhost虚拟环境名称,同样在配置文件中需要使用
```

设置权限：给这个用户所有权限

```
rabbitmqctl set_permissions -p myvhost username01 ".*" ".*" ".*"
```

在"安全组设置中"暴露出端口号

退出后重启

```
docker restart myrabbit
```

在settings.py中指定 消息队列的路径,和结果路径

2.5 版本以后

```
BROKER_URL='amqp://hnq:123456@116.62.155.103:5672/myvhost'
CELERY_RESULT_BACKEND='amqp://hnq:123456@116.62.155.103:5672/myvhost'
```

> **格式**:'amqp://用户名:密码@IP:端口号/虚拟环境'

在2.5 版本之前

```
BROKER_HOST = "116.62.155.103" # IP
BROKER_PORT = 5672 # 端口号
BROKER_USER = "hnq" # 用户名
BROKER_PASSWORD = "123456" # 密码
BROKER_VHOST = "/myvhost" # 虚拟环境
```