Redis,reaatibmq,celery,sqlalchemy实用文档

https://zwdtuser.sh.gov.cn/uc/login/login.jsp?self=self&type=1&jump=&redirect\_uri=http://yct.sh.gov.cn/portal\_yct/webportal/handle\_progress.do

Sqlalchemy:

Celery/celery\_once:

-11.关于to\_analysis.apply\_async(args=[name], retry=**True**, queue=**'to\_analysis'**,immutable=**True**)的参数作用

-10.判断任务是否执行完毕：

>>> result = add.delay(2, 6)

>>> result.ready() # 使用 ready() 判断任务是否执行完毕

False

>>> result.ready()

False

>>> result.ready()

True

>>> result.get() # 使用 get() 获取任务结果

8

-9.前后台启动的多个模式：

celery -A CeleryPro worker -loglevel=info # 前台启动不推荐

celery -A CeleryPro worker -l info # 前台启动简写

celery multi start w1 -A CeleryPro -l info # 推荐用后台启动

W1是worker的名称

-8.基本命令对应的函数和模块：celery -A tasks worker -l info -P solo

Tasks 是tasks.py

如果是多队列的话则 -Q 是执行任务的函数名。

app\_one= Celery(**'tasksa'**,broker=**'amqp://guest@localhost//'**,backend=**'redis://:123456@localhost:6379/0'**)

在这个一般tasksa即是celery的名字，实际情况下写tasks.py的名字就行

-A 的作用：

指的是当前项目

-7.celery 分布式消息队列的实际工作原理

从中可以明白celery的真实作用是delay发送任务放到所谓的rabbtmq中，启动的worker相当于是celery自己写的调度程序，一般默认启动四个工人，有专门的任务调度模式去调度工人从rabbitmq的消息队列中拿任务，然后这个任务就是注册的.task()函数，完成以后返回到backend的redis中。即任务成功。所以假设需要进行做高速缓存服务的话，还是需要引入redis的包然后写存数据到redis 的缓存数据库中。

-6.在redis可视化界面中的结果显示：

{

"status": "FAILURE",

"result": {

"exc\_type": "NotRegistered",

"exc\_message": [

"slow\_task"

],

"exc\_module": "celery.exceptions"

},

"traceback": null,

"children": [],

"task\_id": "0194d4f5-f612-4ef8-a921-a816d0d3e086",

"date\_done": "2019-05-04T11:26:54.032883"

}

-5. -l 的作用：

就是执行该任务下面所有的注册的任务

一般执行任务程序代码如下所示：

app\_one= Celery(**'tasks'**,broker=**'amqp://guest@localhost//'**,backend=**'redis://:123456@localhost:6379/0'**)

@app\_one.task()  
**def** add(x,y):  
 print(1000,x,y)print(**'dome'**)  
 **return "Done!"**@app\_one.task()  
**def** addt(x, y):  
 print(2000, x, y)print(**'dome'**)  
 **return "Done!"**

一般任务注册代码;

**from** tasks **import** add,addt  
  
result=add.delay(1,2)  
print(result.get(timeout=2)) *res:done*result=addt.delay(2,3)  
print(result.get(timeout=1))

Res:done

-4.-Q 的作用：

如果是-Q的话则是多work多队列的形式这个需要在配置文件里面另外再做配置才行

-3.celery的三种启动模式：

celery -A celery\_server worker --loglevel=info

# 启动worker服务了，此后该worker会一直等待任务并执行, 这种方式默认是多进程启动worker

celery -A celery\_server worker --loglevel=info -P solo

使用单进程启动

celery -A celery\_server worker --loglevel=info -P eventlet

使用协程的方式启动。

-2.celery中给redis 的url添加密码：

**'url'**: **'redis://:123456@localhost:6379/0'**,

-1.celery的启动命令：

红色的是py文件 auto\_work是注册的任务

Window10不支持4版本以上celery 所有要加-P=eventlet

Eventlet需要通过pip install 安装才行

celery -A celery\_autowork\_task worker --loglevel=info -Q auto\_work -P=eventlet

1. celery的启动：

安装celery库，将路径python3.7/scrips添加到系统路径下

1.Broker\_url的用法

broker='amqp://guest@localhost// 这是celery的默认broker\_url

即官方推荐的默认消息队列是rabbitmq

broker\_url = 'amqp://guest:guest@localhost:5672//'

基础配置值CELERY--ONCE：

# 一般之前的配置没有这个，需要添加上

celery.conf.ONCE = {

'backend': 'celery\_once.backends.Redis',

'settings': {

'url': 'redis://localhost:6379/0',

'default\_timeout': 60 \* 60

}

}

这里redis的默认端口号就是6379

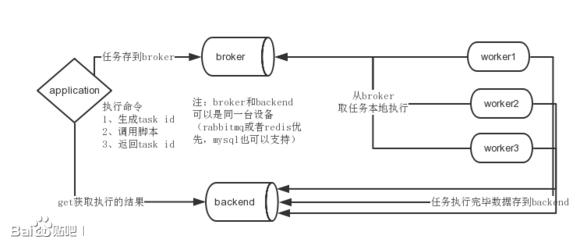
真正痛苦的地方：

result=addt.apply\_async(args=(111,2), countdown=10)

Dely的一种封装形式：countdown是隔10秒以后在执行

Reids:  
一:简单说明一下celery分别在定时任务和异步任务时候的功能  
定时任务的时候  
生产环境经常会跑一些定时任务。假如你有上千台的服务器、上千种任务，定时任务的管理很困难，Celery可以帮助我们快速在不同的机器设定不同种任务。  
异步任务的时候  
celery的broker监听消息队列，从消息队列中分配worker去执行异步任务。最后返回任务的执行状态和结果。

二:简单叙述一下如图的工作流程

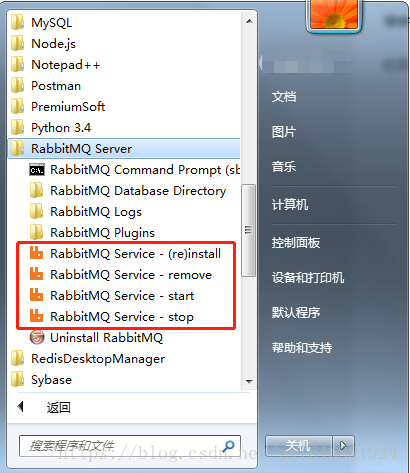


注册一个celery任务，将他封装成一个app。  
启动一个broker，监听消息  
调用任务函数，生成task\_id到消息队列中，  
broker监听到消息信号以后从消息队列中取出任务，调度woker去执行任务  
任务一旦完成将数据存入后端的数据库  
最后返回任务的执行状态和结果

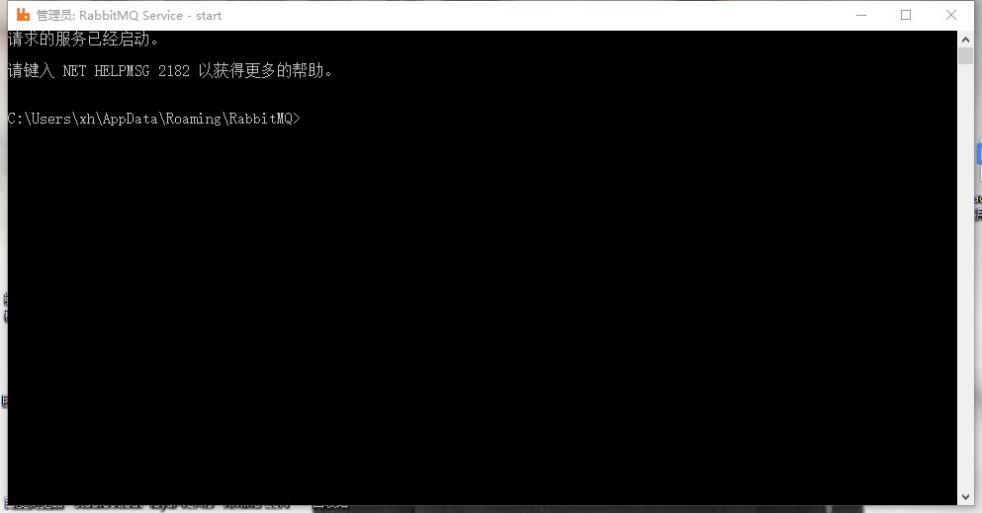
3.如何判断redis服务启动成功  
在redis的根路径下启动powershell执行redis的启动命令  
.\redis-server.exe redis.windows.conf --maxmemory 200M  
启动redis-cli.exe 输入auth 123456 返回ok即redis服务启动成功

rabbitmq:

1. Rabbitmq启动方式：



点击rabbitmq\_service\_start返回



2.访问浏览器的可视化界面

当启动rabbitmq的服务以后输入http://127.0.0.1:15672/即可

默认的账号密码都是guest

