#进阶用法

1.1 Celery异步函数回调

经过快速入门的学习后，我们已经能够使用 Celery 管理普通任务，但对于实际使用场景来说这是远远不够的，所以我们需要更深入的去了解 Celery 更多的使用方式。

首先来看之前的task：

In [3]:
```
import time
from celery import Celery, platforms
platforms.C_FORCE_ROOT = True


from celery.app.task import Task

broker = 'redis://127.0.0.1:6379'
backend = 'redis://127.0.0.1:6379'

app = Celery(__file__, broker=broker, backend=backend)


@app.task
def add(x, y):
    time.sleep(2)
    return  x / y
```

这里的装饰器app.task实际上是将一个正常的函数修饰成了一个celery task对象，所以这里我们可以给修饰器加上参数来决定修饰后的task对象的一些属性，我们也可以自己复写task类然后让这个自定义task修饰函数add，来做一些自定义操作，比如celery修饰的函数执行成功 失败 执行完毕时的行为.

In [ ]:
```
import time
from celery import Celery, platforms
platforms.C_FORCE_ROOT = True


from celery.app.task import Task

broker = 'redis://127.0.0.1:6379'
backend = 'redis://127.0.0.1:6379'

app = Celery(__file__, broker=broker, backend=backend)


class Mytask(Task):

    def on_success(self, retval, task_id, args, kwargs):
        print 'task success 11111'
        return super(Mytask, self).on_success(retval, task_id, args, kwargs)

    def on_failure(self, exc, task_id, args, kwargs, einfo):
        print 'task failed'
        return super(Mytask, self).on_failure(exc, task_id, args, kwargs, einfo)

    def after_return(self, status, retval, task_id, args, kwargs, einfo):
        print 'this is after return'
        return super(Mytask, self).after_return(status, retval, task_id, args, kwargs, einfo)

    def on_retry(self, exc, task_id, args, kwargs, einfo):
        print 'this is retry'
        return super(Mytask,self).on_retry(exc, task_id, args, kwargs, einfo)


@app.task(base=Mytask)
def add(x, y):
    time.sleep(2)
    return  x / y
```

```
celery -A tasks worker --loglevel=info   ##启动任务



In [2]: t = add.delay(1, 2)  ##执行celery函数

在worker中会有类似如下输出：
[2018-10-19 15:05:18,986: WARNING/Worker-1] task success 11111
[2018-10-19 15:05:18,987: WARNING/Worker-1] this is after return
[2018-10-19 15:05:18,988: INFO/MainProcess] Task task.add[c41343cf-e5ca-49f7-9478-63f4c2e2797f] succeeded in 2.01661233298s: 0


In [3]: t = add.delay(1, 0) ##执行celery函数，此函数会由于函数代码中分母为0而报错.

在worker中会有类似如下输出：
[2018-10-19 15:13:38,260: WARNING/Worker-2] task failed
[2018-10-19 15:13:38,261: WARNING/Worker-2] this is after return
[2018-10-19 15:13:38,262: ERROR/MainProcess] Task task.add[12a852cd-726e-44a0-97a8-eb17308c354e] raised unexpected: ZeroDivisionError('integer division or modulo by
zero',)
Traceback (most recent call last):
  File "/app_shell/websocket/lib/python2.7/site-packages/celery/app/trace.py", line 240, in trace_task
    R = retval = fun(*args, **kwargs)
  File "/app_shell/websocket/lib/python2.7/site-packages/celery/app/trace.py", line 438, in __protected_call__
    return self.run(*args, **kwargs)
  File "/app_shell/websocket/demo/task.py", line 38, in add
    return  x / y
ZeroDivisionError: integer division or modulo by zero
```

1.2 将修饰函数成为Task类的绑定方法，执行中的任务获取到了自己执行任务的各种信息，可以根据这些信息做很多其他操作.

In [ ]:
```
import time
from celery import Celery, platforms
platforms.C_FORCE_ROOT = True


from celery.app.task import Task

broker = 'redis://127.0.0.1:6379'
backend = 'redis://127.0.0.1:6379'

app = Celery(__file__, broker=broker, backend=backend)

@app.task(base=Mytask, bind=True)
def add(self, x, y):
    print self.request
    time.sleep(2)
    return  x / y
```

```
celery -A tasks worker --loglevel=info   ##启动任务

In [2]: t = add.delay(1, 2)

在worker中会有类似如下输出：
```

```
[2018-10-19 15:05:16,972: WARNING/Worker-1] <Context: {'chord': None, 'retries': 0, 'args': (1, 2), u'is_eager': False, u'correlation_id': u'c41343cf-e5ca-49f7-9478-63f4c2e2797f', 'errbacks': None, 'taskset': None, 'id': 'c41343cf-e5ca-49f7-9478-63f4c2e2797f', u'headers': {}, 'called_directly': False, 'utc': True, 'task': 'task.add', u'group': None, 'callbacks': None, u'delivery_info': {u'priority': 0, u'redelivered': None, u'routing_key': u'celery', u'exchange': u'celery'}, u'hostname': 'celery@gitlab.example.com', 'expires': None, 'timelimit': (None, None), 'eta': None, 'kwargs': {}, u'reply_to': u'46af07e6-8e03-3574-9608-fc8c0b10a98e', '_protected': 1}>
[2018-10-19 15:05:18,986: WARNING/Worker-1] task success 11111
[2018-10-19 15:05:18,987: WARNING/Worker-1] this is after return
```

In [ ]: