day11 拍卖

今日概要

- celery，处理任务的Python的模块。
  - 场景1：

    对【耗时的任务】，通过celery，将任务添加到broker（队列），然后立即给用户返回一个任务ID。
    当任务添加到broker之后，由worker去broker获取任务并处理任务。
    任务弯完成之后，再将结果放到backend中

    用户想要检查结果，提供任务ID，我们就可以去backend中去帮他查找。

  - 场景2：

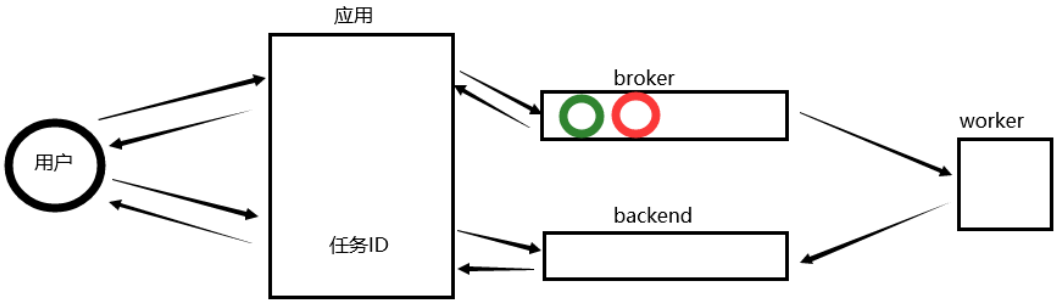
    定时任务（定时发布/定时拍卖）

- 拍卖系列接口

今日详细

1.celery

celery是一个基于Python开发的模块，可以帮助我们对任务进行分发和处理。



1.1 环境的搭建

```
pip3 install celery==4.4
安装broker: redis或rabbitMQ
pip3 install redis / pika
```

1.2 快速使用

- s1.py

```
from celery import Celery

app = Celery('tasks', broker='redis://192.168.10.48:6379',
backend='redis://192.168.10.48:6379')

@app.task
def x1(x, y):
    return x + y

@app.task
def x2(x, y):
    return x - y
```

- s2.py

```
from s1 import x1

result = x1.delay(4, 4)
print(result.id)
```

- s3.py

```
from celery.result import AsyncResult
from s1 import app

result_object = AsyncResult(id="任务ID", app=app)
print(result_object.status)
```

运行程序：

1. 启动redis
2. 启动worker

```
# 进入当前目录
celery worker -A s1 -l info
```

```
windows

Traceback (most recent call last):
  File "d:\wupeiqi\py_virtual_envs\auction\lib\site-
packages\billiard\pool.py", line 362, in workloop
    result = (True, prepare_result(fun(*args, **kwargs)))
  File "d:\wupeiqi\py_virtual_envs\auction\lib\site-
packages\celery\app\trace.py", line 546, in _fast_trace_task
    tasks, accept, hostname = _loc
ValueError: not enough values to unpack (expected 3, got 0)

pip install eventlet

celery worker -A s1 -l info -P eventlet
```

3. 创建任务

```
python s2.py
python s2.py
```

4. 查看任务状态

```
# 填写任务ID
ptyhon s3.py
```

1.3 django中应用celery

之后，需要按照django-celery的要求进行编写代码。

- 第一步：【项目/项目/settings.py 】添加配置

```
CELERY_BROKER_URL = 'redis://192.168.16.85:6379'
CELERY_ACCEPT_CONTENT = ['json']
CELERY_RESULT_BACKEND = 'redis://192.168.16.85:6379'
CELERY_TASK_SERIALIZER = 'json'
```

- 第二步：【项目/项目/celery.py】在项目同名目录创建 celery.py

```
#!/usr/bin/env python
# -*- coding:utf-8 -*-

import os
from celery import Celery

# set the default Django settings module for the 'celery' program.
os.environ.setdefault('DJANGO_SETTINGS_MODULE', 'demos.settings')

app = Celery('demos')

# Using a string here means the worker doesn't have to serialize
# the configuration object to child processes.
# - namespace='CELERY' means all celery-related configuration keys
#   should have a `CELERY_` prefix.
app.config_from_object('django.conf:settings', namespace='CELERY')

# Load task modules from all registered Django app configs.
# 去每个已注册app中读取 tasks.py 文件
app.autodiscover_tasks()
```

- 第三步，【项目/app名称/tasks.py】

```
from celery import shared_task

@shared_task
def add(x, y):
    return x + y

@shared_task
def mul(x, y):
    return x * y
```

- 第四步，【项目/项目/`__init__.py`】

```
from .celery import app as celery_app

__all__ = ('celery_app',)
```

- 启动worker

```
进入项目目录

celery worker -A demos -l info -P eventlet
```

- 编写视图函数，调用celery去创建任务。
  - url

```
url(r'^create/task/$', task.create_task),
url(r'^get/result/$', task.get_result),
```

  - 视图函数

```
from django.shortcuts import HttpResponse
from api.tasks import x1

def create_task(request):
    print('请求来了')
    result = x1.delay(2,2)
    print('执行完毕')
    return HttpResponse(result.id)


def get_result(request):
    nid = request.GET.get('nid')
    from celery.result import AsyncResult
    # from demos.celery import app
    from demos import celery_app
    result_object = AsyncResult(id=nid, app=celery_app)
    # print(result_object.status)
    data = result_object.get()
    return HttpResponse(data)
```

- 启动django程序

```
python manage.py ....
```

1.4 celery定时执行

```

```

1.5 周期性定时任务

- celery
- django中也可以结合使用

2.拍卖业务

2.1 表结构

2.2 数据初始化

2.3 接口

- 专场列表
- 专场详细
- 单品详细
- 保证金页面
    - 单品信息
    - 保证金
        - 全场
        - 单品
    - 支付方式
        - 微信
        - 余额
- 竞价
    - 查看
    - 提交

其他（2月3号）

1. 支付
2. 优惠券
3. 退款处理
4. 项目部署（小程序/api接口）