day03 定时任务&订单

内容回顾

- 项目架构
  - 小程序
  - API
  - 后台管理
    - 专场 & 拍品管理
    - 订单（定时任务）
- 业务流程
  - 运营人员
    - 后台管理创建 专场 & 拍品 信息（待发布）
    - 定时任务来进行 专场 & 拍品 信息更新和处理
      - 待发布 -> 预展
      - 预展 -> 拍卖中
      - 拍卖中 -> 拍卖结束
    - 优惠券管理
  - 用户
    - 看到拍卖的预展商品信息
    - 进行竞价拍卖
    - 支付 + 地址 + 优惠券

今日概要

- celery继承在django中
- 业务流程
  - 待发布 -> 预展
  - 预展 -> 拍卖中
  - 拍卖中 -> 拍卖结束（表结构+处理思路）

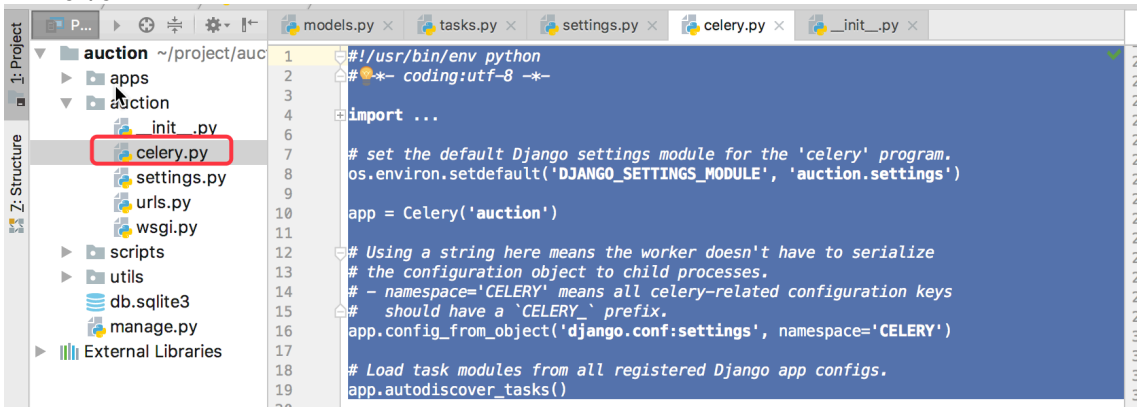
今日详细

1. django集成Celery

- 启动redis
- django中配置celery
  - settings.py

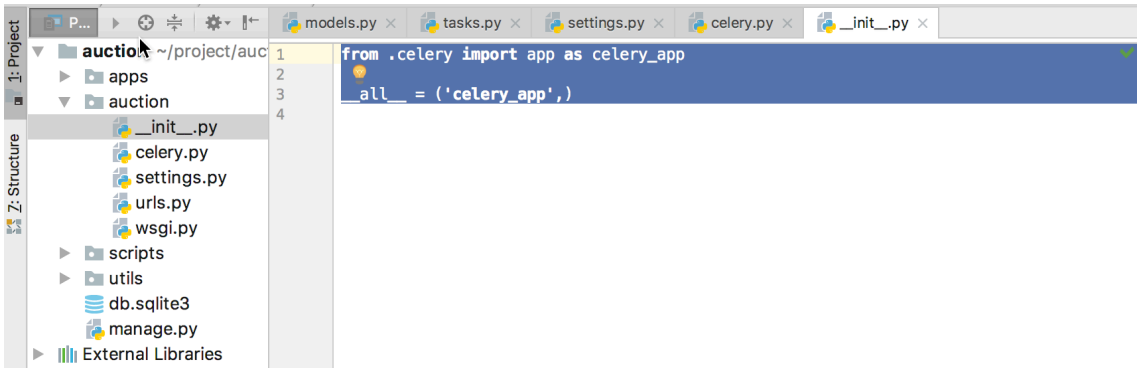
```
CELERY_BROKER_URL = 'redis://10.211.55.25:6379'
CELERY_ACCEPT_CONTENT = ['json']
CELERY_RESULT_BACKEND = 'redis://10.211.55.25:6379'
CELERY_TASK_SERIALIZER = 'json'
```

- celery.py




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

import os
from celery import Celery

os.environ.setdefault('DJANGO_SETTINGS_MODULE', '项目名称.settings')

app = Celery('项目名称')

app.config_from_object('django.conf:settings', namespace='CELERY')

app.autodiscover_tasks()
```

- `__init__.py`




```
from .celery import app as celery_app

__all__ = ('celery_app',)
```

- 在每个app中创建tasks.py

```
#!/usr/bin/env python
# -*- coding:utf-8 -*-
import uuid
from celery import shared_task
from apps.api import models

@shared_task
def to_preview_status_task(auction_id):
    pass

@shared_task
def to_auction_status_task(auction_id):
    pass

@shared_task
def end_auction_task(auction_id):
    pass
```

- 视图函数中调用

```
def auction_add(request):
    result = tasks.to_preview_status_task.delay(1)
    print(result.id)

    result = tasks.to_preview_status_task.apply_async(args=[1], eta=什么时候执行UTC)
    print(result.id)
```

- 项目启动

```
celery worker:   celery workder -A auction -l info
django项目:       python manage.py runserver
```

- 用户访问

扩展：django项目启动时加载文件

- 启动django
- 用户来请求

大家在使用django admin时，会有这样的现象。

- 如果在项目中运行，那么在django加载时就回去每个app中调用xx.py文件

```
from django.utils.module_loading import autodiscover_modules
autodiscover_modules('xx')
```

- django源码的启动流程

```
[读取配置文件加载app] -> [autodiscover_modules(xx.py)] -> [加载文件]
```

```
from django.apps import AppConfig

class ApiConfig(AppConfig):
    name = 'apps.api'

    def ready(self):
        super().ready()
        from django.utils.module_loading import autodiscover_modules
        autodiscover_modules('xx')
```

任务：在自己的项目中集成celery

2.业务开发

2.1 创建专场，3个定时任务


```
class Auction(models.Model):
    """
    拍卖系列
    """
    title = models.CharField(verbose_name='标题', max_length=32)
    status_choices = (
        (1, '未开拍'),
        (2, '预展中'),
        (3, '拍卖中'),
        (4, '已结束')
    )
    status = models.PositiveSmallIntegerField(verbose_name='状态',
choices=status_choices, default=1)

    cover = models.FileField(verbose_name='封面', max_length=128)
    video = models.CharField(verbose_name='预览视频', max_length=128,
null=True, blank=True)
```

```
    preview_start_time = models.DateTimeField(verbose_name='预展开始时间')
    preview_end_time = models.DateTimeField(verbose_name='预展结束时间')

    auction_start_time = models.DateTimeField(verbose_name='拍卖开始时间')
    auction_end_time = models.DateTimeField(verbose_name='拍卖结束时间')

    deposit = models.PositiveIntegerField(verbose_name='全场保证金',
default=1000)

    total_price = models.PositiveIntegerField(verbose_name='成交额', null=True,
blank=True)
    goods_count = models.PositiveIntegerField(verbose_name='拍品数量',
default=0)
    bid_count = models.PositiveIntegerField(verbose_name='出价次数', default=0)
    look_count = models.PositiveIntegerField(verbose_name='围观次数',
default=0)
    create_time = models.DateTimeField(verbose_name='创建时间',
auto_now_add=True)

    class Meta:
        verbose_name_plural = '拍卖系列'

    def __str__(self):
        return self.title


class AuctionTask(models.Model):
    """ 定时任务 """
    auction = models.OneToOneField(verbose_name='专场', to='Auction')

    preview_task = models.CharField(verbose_name='Celery预展任务ID',
max_length=64)

    auction_task = models.CharField(verbose_name='Celery拍卖任务ID',
max_length=64)

    auction_end_task = models.CharField(verbose_name='Celery拍卖结束任务ID',
max_length=64)
```

```
def auction_add(request):
    """
    创建拍卖任务
    :param request:
    :return:
    """
    if request.method == 'GET':
        form = AuctionModelForm()
```

```
        return render(request, 'web/auction_form.html', {'form': form})
    form = AuctionModelForm(data=request.POST, files=request.FILES)

    if form.is_valid():
        # 1.专场信息写入数据库，在Auction表中添加了一条数据。
        instance = form.save()

        # 2.创建定时任务

        # 1.定时任务，从 未开拍  预展
        preview_utc_datetime =
datetime.datetime.utcfromtimestamp(instance.preview_start_time.timestamp())
        preview_task_id = tasks.to_preview_status_task.apply_async(args=
[instance.id], eta=preview_utc_datetime).id

        # 2.定时任务，从 预展 到 开拍
        auction_utc_datetime =
datetime.datetime.utcfromtimestamp(form.instance.auction_start_time.timestamp(
))
        auction_task_id = tasks.to_auction_status_task.apply_async(args=
[instance.id], eta=auction_utc_datetime).id

        # 3.定时任务，从 开拍 到 拍卖结束
        auction_end_utc_datetime =
datetime.datetime.utcfromtimestamp(form.instance.auction_end_time.timestamp())
        auction_end_task_id = tasks.end_auction_task.apply_async(args=
[instance.id], eta=auction_end_utc_datetime).id

        models.AuctionTask.objects.create(
            auction=instance,
            preview_task=preview_task_id,
            auction_task=auction_task_id,
            auction_end_task=auction_end_task_id
        )

        return redirect('auction_list')
    return render(request, 'web/auction_form.html', {'form': form})
```

```
import uuid
from celery import shared_task
from apps.api import models


@shared_task
def to_preview_status_task(auction_id):
    models.Auction.objects.filter(id=auction_id).update(status=2)
    models.AuctionItem.objects.filter(auction_id=auction_id).update(status=2)
```

```python
@shared_task
def to_auction_status_task(auction_id):
    models.Auction.objects.filter(id=auction_id).update(status=3)
    models.AuctionItem.objects.filter(auction_id=auction_id).update(status=3)

@shared_task
def end_auction_task(auction_id):
    models.Auction.objects.filter(id=auction_id).update(status=4)
    models.AuctionItem.objects.filter(auction_id=auction_id).update(status=4)
    # 之后应该做什么？ 表结构设计 + 实现思路
```

任务：简单定时任务集成项目中

任务：拍卖结束时应该做什么？

```
表结构设计 + 实现思路
```

为了方便咱沟通，接下来项目设计和实现环节去QQ群通过群语音。

1. 打开全栈24期QQ群语音。
2. 进入群语音指挥 默认将自己的 麦 关闭，以免声音嘈杂。
3. 有问题可以自行打开麦进行说话。

3.答疑

3.1 celery中task和share_task装饰器的区别?

```python
import time
from celery import Celery

app =
Celery('tasks',broker='redis://192.168.10.48:6379',backend='redis://192.168.10
.48:6379')

@app.task
def xxxxxx(x, y):
    time.sleep(10)
    return x + y
```

装饰函数，将函数当做celery的任务函数。

```python
import time
from celery import Celery
from celery import shared_task
```

```
app =
Celery('tasks',broker='redis://192.168.10.48:6379',backend='redis://192.168.10
.48:6379')

@shared_task
def xxxxxx(x, y):
    time.sleep(10)
    return x + y
```

装饰函数，将函数当做celery的任务函数。
不依赖某个celery对象，而是加载到内存之后自动添加到celery对象中。
s1.py

```
    import time
    from celery import Celery
    app = Celery('tasks',broker='r...')

s2.py
    from celery import shared_task

    @shared_task
    def x1(x, y):
        time.sleep(10)
        return x + y
s3.py
    import s1
    import s2
```

```
import time
from celery import Celery
from celery import shared_task

app1 = Celery('t1',broker='redis://192.168.10.48:6379')
app2 = Celery('t2',broker='redis://192.168.10.49:6379')

@app1.task
def x1(x, y):
    time.sleep(10)
    return x + y

@app2.task
def x2():
    return 666

@shared_task
def x3():
    pass
```

share_task作用

- 不依赖celery对象，加载内存之后自动关联celery对象（*）。
- 于多个celery对象进行关联。

3.2 CharField & FileField的区别?

django的Model类中CharField & FileField的区别有哪些?

```
class David(models.Model):
    title = models.CharField(....)
    img = models.FileField(...,upload_to="media")
```

- 数据库中的字段

  ```
  title -> 字符串
  img   -> 字符串
  ```

- ModelForm对数据库进行操作时不同。

  FileField字段会生成 file 标签，并且在提交时会自动将文件上传到服务器上，同时将文件路径保存到数据库。

  ```
  class DavidModelForm(ModelForm):
      class Meta:
          model = models.David
          fields = "__all__"

  def index(request):
      if request.method == "GET":
          form = DavidModelForm()
          return render(request,'index.html',{'form':form})
      form = DavidModelForm(data=request.POST,files=rqeuest.Files)
      # 如果是FileField字段，则自动会将上传的文件到项目根目录。
      form.save()
  ```

  ```
  # html
  form.title  ->  input text标签
  form.img    ->  input file标签
  ```

扩展：想法在数据库

```
class David(models.Model):
    title = models.CharField(....)
    img = models.CharField()
```

```
class DavidModelForm(ModelForm):
    class Meta:
        model = models.David
        fields = "__all__"
        widgets = {
            'img': FileField
        }

    def clean_img(self):
        自定义shangchan
        return '路径'
```

3.3 URLField和ImageField的区别?

```
URLField
    在数据库中存储的字符串，在ModelForm进行页面操作时内置了正则表达式。
ImageField & FileFied 相似
    在数据库中存储的字符串，在ModelForm进行页面操作 input file标签.
    ImageField在内部集成PIL模块调用，他可以获取图片的尺寸。
```

更多可参考这里：https://www.cnblogs.com/wupeiqi/articles/6216618.html

4.今日任务

4.1 项目中集成celery

4.2 三个定时任务基本

4.3 拍卖结束时应该做什么?

- 刘兵

  ```
  生成订单
  给交保证金的用户返还保证金
  ```

- 子凡

  ```
  获取本次拍卖专场中的所有单品
      每一件单品获取他们的出价最高者，为他生成一个订单：状态待付款
      对此订单再创建一个定时任务24小时之后，检查是否付款，没付款，扣除保证金。

      拍品对象：
          成交价
          状态，成功（逾期未付款）。

      没有拍到的要退还保证金
  ```

- 周金刚

  用户未付款，成交价。
  更新专场的状态，计算总成交额。

- 武沛齐
  - 数据关系

```
专场1(名称、封面、状态、总成交额、专场保证金1000)
    拍品A(状态、成交价即最高价、单品保证金200)
        500 白浩伟
        400 华新
        300 朱凡宇
        200 华新
        100 白浩伟
    拍品B(状态、成交价即最高价、单品保证金900)
        10000 白浩伟
        200 朱凡宇
        150 白浩伟
        50  白浩伟
    拍品C(状态、成交价即最高价、单品保证金100)

保证金
    白浩伟      1000     专场1
    华新        200      专场1     拍品A
    朱凡宇      200      专场1     拍品A
    朱凡宇      100      专场1     拍品C
```

  - 逻辑处理

```
auction_object = 获取专场对象
item_object_list = 获取对应专场下的所有拍品(专场=auction_object)

1.专场状态的更新
    auction_object.status = "拍卖结束"
    auction_object.save()
2.拍品 状态 更新
    total_成交价 = 0
    for item_object in item_object_list:
      2.1 检查拍品没有人出价，则更新 item_object.status = "流派"

      2.2 有人出价则找到出价最高者
          lk=models.出价记录表.objects.filter(拍品
=item_object).order_by('-出价').first()

          item_object.成交价 = lk.出价
          item_object.状态 = 成交
          total_成交价 += lk.出价
```

```
      2.3 为出价高者创建订单
        order_object = models.订单.objects.create(
            订单号=随机字符串
            状态=待支付,
            拍品=item_object,
            用户=lk.用户,
            价格=lk.出价,
            保证金=当前这个人为这件商品付的保证金对象(专场/单品)
        )
      2.4 找到未拍到商品的人退还保证金(人去重)
        单品保证金:
            立即退换保证金
        全场的保证金:
            他没有拍到当前专场的其他拍品，立即退还保证金。
            拍到其他拍品，保证金冻结。

      2.5 调用一个定时任务 24小时之后执行。
        24小时任务.apply_async(arg=[order_object.id],eta=当前时间+24小时)
    auction_object.总成交额 = total_成交价
    auction_object.save()


@shared_task
def 24小时任务(订单id):
    如果订单已经支付，则不再处理。
    订单未支付，则需要处理(逾期未付款)
    order_object = models.订单.objects.filter(id=订单id).first()
    1. 订单状态更新
        order_object.status = "逾期未付款"
        order_object.save()
    2. 拍品状态更新
        order_object.拍品.status = "逾期未付款"
        order_object.拍品.save()
    3. 扣除保证金
        # 200
        if order_object.保证金.类型 == '单品':
            直接扣除保证金
            order_object.保证金.balance余额 = 0
            order_object.保证金.save()
        else: order_object.保证金.类型 == '专场保证金':
            如果 order_object.保证金.balance余额 <= order_object.拍品.保证
金:
                order_object.保证金.balance余额 = 0
                order_object.保证金.save()
            else: 余额还多
                order_object.保证金.balance余额 -= order_object.拍品.保证
金
                order_object.保证金.save()
```

如果 当前用户在此专场中没有其他拍到的拍品，退还余额：
order_object.保证金.balance余额 按照原路返回。