

W-D

起风的日子

博客园

首页

新随笔

联系

订阅

管理

随笔 - 102 文章 - 0 评论 - 67 阅读 - 64万

昵称： W-D
园龄： 5年9个月
粉丝： 240
关注： 36
+加关注

< 2022年9月 >						
日	一	二	三	四	五	六
28	29	30	31	1	2	3
4	5	6	7	8	9	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28	29	30	1
2	3	4	5	6	7	8

随笔分类

Ansible(2)
devops(5)
django(13)
Django Rest Framework(9)
Docker(7)
flask(8)
Golang(10)
html(4)
javascript(6)
jenkins(1)
kubernetes(4)
Load Balance(2)
nginx(2)
python(28)
redis(4)
更多

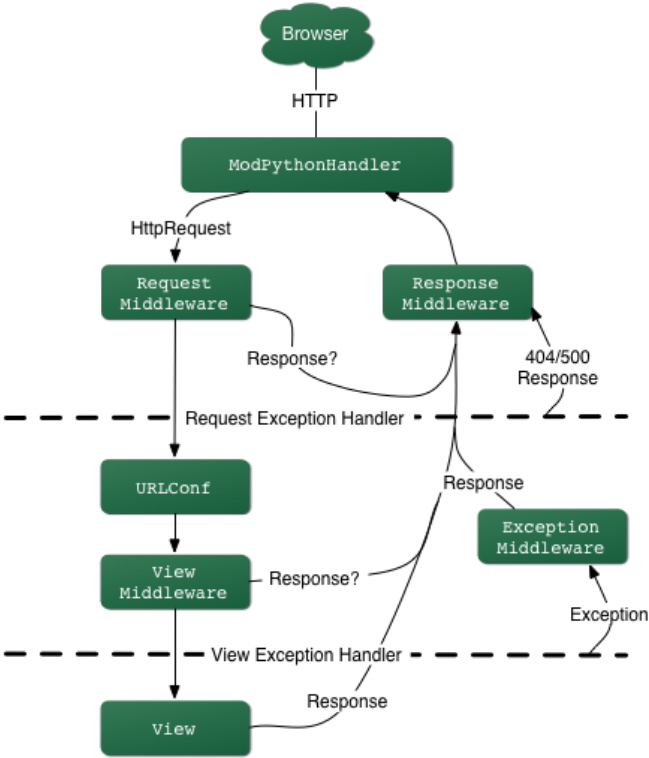
阅读排行榜

1. python中的Queue(队列)详解(70679)
2. Django中使用Celery(57019)

Django中使用Celery

一、前言

Celery是一个基于python开发的分布式任务队列，如果不了解请阅读笔者上一篇博文[Celery入门与进阶](#)，而做python WEB开发最为流行的框架莫属Django，但是Django的请求处理过程都是同步的无法实现异步任务，若要实现异步任务处理需要通过其他方式（前端的一目录决方案是ajax操作），而后台Celery就是不错的选择。倘若一个用户在执行某些操作需要等待，才返回，这大大降低了网站的吞吐量。下面将描述Django的请求处理大致流程（图片来源于网络）：



请求过程简单说明：浏览器发起请求-->请求处理-->请求经过中间件-->路由映射-->视图处理业务逻辑-->响应请求(template或response)

二、配置使用

celery很容易集成到Django框架中，当然如果想要实现定时任务的话还需要安装django-celery-beta插件，后面会说明。需要注意的是Celery4.0只支持Django版本>=1.8的，如果是小于1.8版本需要使用Celery3.1。

配置

新建项目taskproj,目录结构（每个app下多了个tasks文件，用于定义任务）：

```
taskproj
├── app01
│   ├── __init__.py
│   ├── apps.py
│   ├── migrations
│   │   └── __init__.py
│   ├── models.py
│   ├── tasks.py
│   └── views.py
├── manage.py
└── taskproj
```

21

3. go语言之行--golang操作Redis、mysql
大全(43638)
4. go语言之行--golang核武器goroutine
调度原理、channel详解(38959)
5. go语言之行--接口(interface)、反射(reflect)
详解(34462)

- 评论排行榜
1. Django中使用Celery(9)
2. redis系列--你真的入门了吗？ redis4.0
入门~(6)
3. go语言之行--golang核武器goroutine
调度原理、channel详解(6)
4. redis系列--redis4.0深入持久化(5)
5. Docker镜像存储-overlays(3)

- 推荐排行榜
1. Django中使用Celery(21)
2. redis系列--你真的入门了吗？ redis4.0
入门~(17)
3. go语言之行--golang核武器goroutine
调度原理、channel详解(15)
4. 分布式任务队列Celery入门与进阶(14)
5. redis系列--redis4.0深入持久化(13)

```
| | | __init__.py
| | | settings.py
| | | urls.py
| | | wsgi.py
| | | templates
| | |
```

在项目目录taskproj/taskproj/目录下新建celery.py:

```
#!/usr/bin/env python3
# -*- coding:utf-8 -*-
# Author:wd
from __future__ import absolute_import, unicode_literals
import os
from celery import Celery

os.environ.setdefault('DJANGO_SETTINGS_MODULE', 'taskproj.settings') # 设置django环境

app = Celery('taskproj')

app.config_from_object('django.conf:settings', namespace='CELERY') # 使用CELERY_ 作为

app.autodiscover_tasks() # 发现任务文件每个app下的task.py
```

目录
导航

taskproj/taskproj/__init__.py:

```
from __future__ import absolute_import, unicode_literals
from .celery import app as celery_app
__all__ = ['celery_app']
```

taskproj/taskproj/settings.py

```
CELERY_BROKER_URL = 'redis://10.1.210.69:6379/0' # Broker配置，使用Redis作为消息中间件

CELERY_RESULT_BACKEND = 'redis://10.1.210.69:6379/0' # BACKEND配置，这里使用redis

CELERY_RESULT_SERIALIZER = 'json' # 结果序列化方案
```

进入项目的taskproj目录启动worker:

```
celery worker -A taskproj -l debug
```

定义与触发任务

任务定义在每个tasks文件中， app01 / tasks.py:

```
from __future__ import absolute_import, unicode_literals
from celery import shared_task

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

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

视图中触发任务

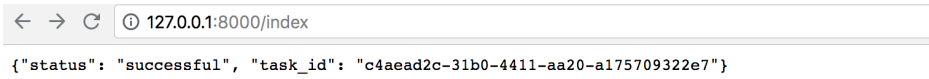
```
from django.http import JsonResponse
from app01 import tasks

# Create your views here.
```

21

```
def index(request,*args,**kwargs):
    res=tasks.add.delay(1,3)
    #任务逻辑
    return JsonResponse({'status':'successful','task_id':res.task_id})
```

访问http://127.0.0.1:8000/index



若想获取任务结果，可以通过task_id使用AsyncResult获取结果,还可以直接通过backend获取：

```
10.1.210.69:6379> get celery-task-meta-c4aead2c-31b0-4411-aa20-a175709322e7
"{\"status\": \"SUCCESS\", \"result\": 4, \"traceback\": null, \"children\": [], \"task_id\": \"c4aead2c-31b0-4411-aa20-a175709322e7\"}"
10.1.210.69:6379>
```

目录
导航

扩展

除了redis、rabbitmq能做结果存储外，还可以使用Django的orm作为结果存储，当然需要安装依赖插件，这样的好处在于我们可以直接通过django的数据查看到任务状态，同时为可以制定更多的操作，下面介绍如何使用orm作为结果存储。

1.安装

```
pip install django-celery-results
```

2.配置settings.py，注册app

```
INSTALLED_APPS = (
    ...,
    'django_celery_results',
)
```

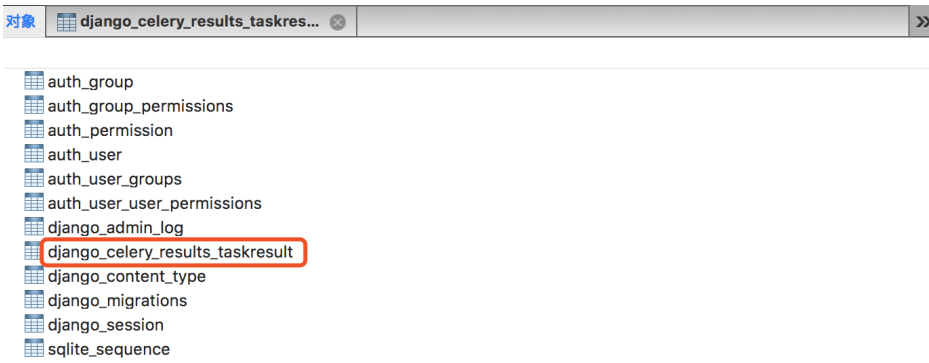
4.修改backend配置，将redis改为django-db

```
#CELERY_RESULT_BACKEND = 'redis://10.1.210.69:6379/0' # BACKEND配置，这里使用redis
CELERY_RESULT_BACKEND = 'django-db' #使用django orm 作为结果存储
```

5.修改数据库

```
python3 manage.py migrate django_celery_results
```

此时会看到数据库会多创建：



当然你有时候需要对task表进行操作，以下源码的表结构定义：

```
class TaskResult(models.Model):
    """Task result/status."""

    task_id = models.CharField(_('task id'), max_length=255, unique=True)
    task_name = models.CharField(_('task name'), null=True, max_length=255)
    task_args = models.TextField(_('task arguments'), null=True)
    task_kwargs = models.TextField(_('task kwargs'), null=True)
```

21

```
status = models.CharField(_('state'), max_length=50,
                           default=states.PENDING,
                           choices=TASK_STATE_CHOICES
                           )

content_type = models.CharField(_('content type'), max_length=128)
content_encoding = models.CharField(_('content encoding'), max_length=64)
result = models.TextField(null=True, default=None, editable=False)
date_done = models.DateTimeField(_('done at'), auto_now=True)
traceback = models.TextField(_('traceback'), blank=True, null=True)
hidden = models.BooleanField(editable=False, default=False, db_index=True)
meta = models.TextField(null=True, default=None, editable=False)

objects = managers.TaskResultManager()

class Meta:
    """Table information."""

    ordering = ['-date_done']

    verbose_name = _('task result')
    verbose_name_plural = _('task results')

def as_dict(self):
    return {
        'task_id': self.task_id,
        'task_name': self.task_name,
        'task_args': self.task_args,
        'task_kwargs': self.task_kwargs,
        'status': self.status,
        'result': self.result,
        'date_done': self.date_done,
        'traceback': self.traceback,
        'meta': self.meta,
    }

def __str__(self):
    return '<Task: {0.task_id} ({0.status})>'.format(self)
```

目录
导航

三、Django中使用定时任务

如果想要在django中使用定时任务功能同样是靠beat完成任务发送功能，当在Django中使用定时任务时，需要安装django-celery-beat插件。以下将介绍使用过程。

安装配置

1.beat插件安装

```
pip3 install django-celery-beat
```

2.注册APP

```
INSTALLED_APPS = [
    ...
    'django_celery_beat',
]
```

3.数据库变更

```
python3 manage.py migrate django_celery_beat
```

4.分别启动woker和beta

```
celery -A proj beat -l info --scheduler django_celery_beat.schedulers:DatabaseScheduler

celery worker -A taskproj -l info #启动woker
```

5.配置admin

urls.py



21

```
# urls.py
from django.conf.urls import url
from django.contrib import admin

urlpatterns = [
    url(r'^admin/', admin.site.urls),
]
```

6.创建用户

```
python3 manage.py createsuperuser
```

7.登录admin进行管理（地址http://127.0.0.1:8000/admin） 并且还可以看到我们上次使用orm作为结果存储的表。

http://127.0.0.1:8000/admin/login/?next=/admin/

Site administration

AUTHENTICATION AND AUTHORIZATION		
Groups	+ Add	Change
Users	+ Add	Change

DJANGO_CELERY_RESULTS		
Task results	任务结果，前提是用了orm做backend	+ Add Change

PERIODIC TASKS		
Crontabs	计划任务定义	+ Add Change
Intervals	间隔任务	+ Add Change
Periodic tasks	任务定义	+ Add Change
Solar events	根据太阳升起、降落订制任务（无用）	+ Add Change

目录导航

Recent actions

My actions

None available

使用示例：

Add periodic task

Name:

test-task

任务名称

Task (registered):

Useful description

app01.tasks.add

app01.tasks.mul

taskproj.celery.debug_task

tasks.py中的任务定义

Task (custom):

☒ Enabled

Schedule

Interval:

every minute

时间间隔

Crontab:

计划任务

Solar:

Use a solar schedule

查看结果:

二次开发

django-celery-beat插件本质上是对数据库表变化检查，一旦有数据库表改变，调度器重新读取任务进行调度，所以如果想自己定制的任务页面，只需要操作beat插件的四张表就可以了。当然你还可以自己定义调度器，django-celery-beat插件已经内置了model，只需要进行导入便可进行orm操作，以下我用django reset api进行示例：

settings.py

urls.py

21

views.py



```
from django_celery_beat.models import PeriodicTask #倒入插件model
from rest_framework import serializers
from rest_framework import pagination
from rest_framework.viewsets import ModelViewSet
class Userserializer(serializers.ModelSerializer):
    class Meta:
        model = PeriodicTask
        fields = '__all__'

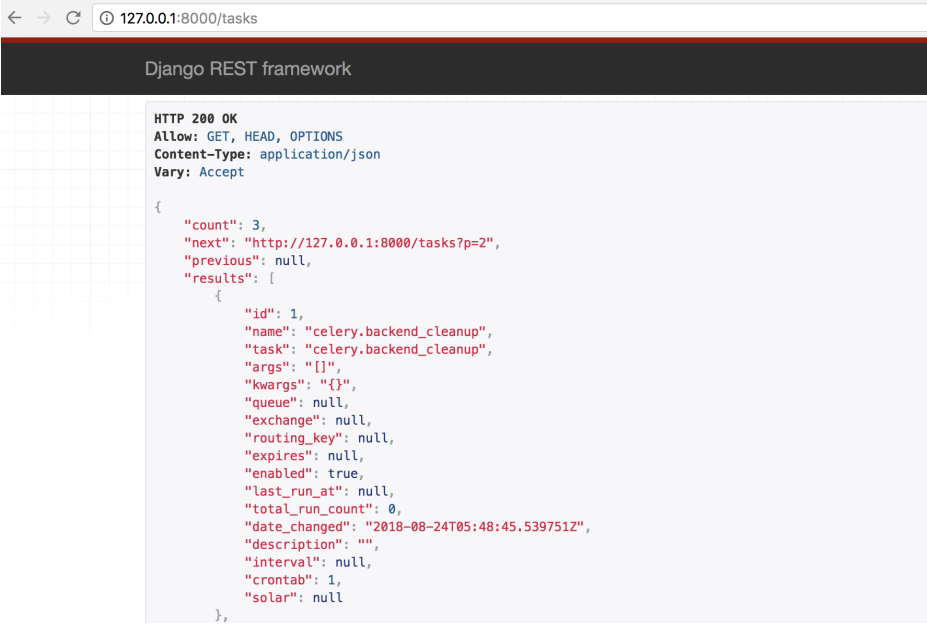
class Mypagination(pagination.PageNumberPagination):
    """自定义分页"""
    page_size=2
    page_query_param = 'p'
    page_size_query_param='size'
    max_page_size=4

class TaskView(ModelViewSet):
    queryset = PeriodicTask.objects.all()
    serializer_class = Userserializer
    permission_classes = []
    pagination_class = Mypagination
```



目录
导航

访问http://127.0.0.1:8000/tasks如下:



分类: [django](#)


好文要顶

关注我

收藏该文







[W-D](#)
粉丝 - 240 关注 - 36

[+加关注](#)

« 上一篇: [Python算法基础](#)
» 下一篇: [redis系列--深入哨兵集群](#)

posted @ 2018-08-24 15:35 W-D 阅读(57019) 评论(9) 编辑 收藏 举报

[刷新评论](#) [刷新页面](#) [返回顶部](#)

登录后才能查看或发表评论, 立即 [登录](#) 或者 [逛逛](#) 博客园首页

- 【推荐】亚马逊云科技现身世界人工智能大会，揭示AI最新趋势
- 【推荐】下一步，敏捷！云可达科技SpecDD敏捷开发专区
- 【推荐】腾讯云多款云产品1折起，买云服务器送免费机器
- 【推荐】天翼云新客特惠，云主机1核2G低至33.43元/年

目录
导航

编辑推荐:

- 记一次 .NET 某打印服务 非托管内存泄漏分析
- 使用 Three.js 实现一个创意纪念页面
- 用自己的编程语言实现了一个网站
- 新时代布局新特性 -- 容器查询
- EntityFrameworkCore 模型自动更新（下）

最新新闻:

- 经纬张颖给技术背景创始人十条建议：野蛮生长时代已结束，只有坚信才会更好
- 重磅！我国首次火星探测任务一批科学研究成果发布，火星上有水吗？答案来了！我国还实现了这个“首次”
- 南开介教授好嘛是“段子手”
- “赢一把我就睡”，“羊了个羊”们有多“上头”？
- 神舟十四号航天员完成舱外救援验证 出舱时机怎样确定？
- » 更多新闻...