ТЕХНОТРЕК

Лекция 5

Очереди и задачи. Celery, redis, cron.

Может лучше сделать что-то асинхронно?

Celery

Celery - распределённая очередь заданий, реализованная на языке Python, служит для хранения отложенных задач

Преимущества:

- выполнение некоторого кода в фоновом режиме
- возможность ускорения времени ответа сервера

```
pip install celery
```

Фоновые задачи

- отправка уведомлений (email, sms, push, desktop)
- периодическое обновление данных
- генерация отчетов

Celery. Основные понятия

- **Брокер** (broker) - служит для передачи сообщений (задач) между так называемыми исполнителями (**workers**). Celery общается с брокером по протоколу AMQP

  - Redis (производительность с celery выше)
  - RabbitMQ

- **Бэкенд** (backend) - хранилище результатов выполнения задач

  - Memcached

Поднимаем redis
redis-server /usr/local/etc/redis.conf

Конфигурируем

https://docs.celeryproject.org/en/latest/django/first-steps-with-django.html

в качестве бэкенда используем Redis

Пишем первый таск

```
from application.celery_app import app


@app.task()
def add_together(a, b):
    return a + b
```

Запускаем celery. Отправляем таск в очередь

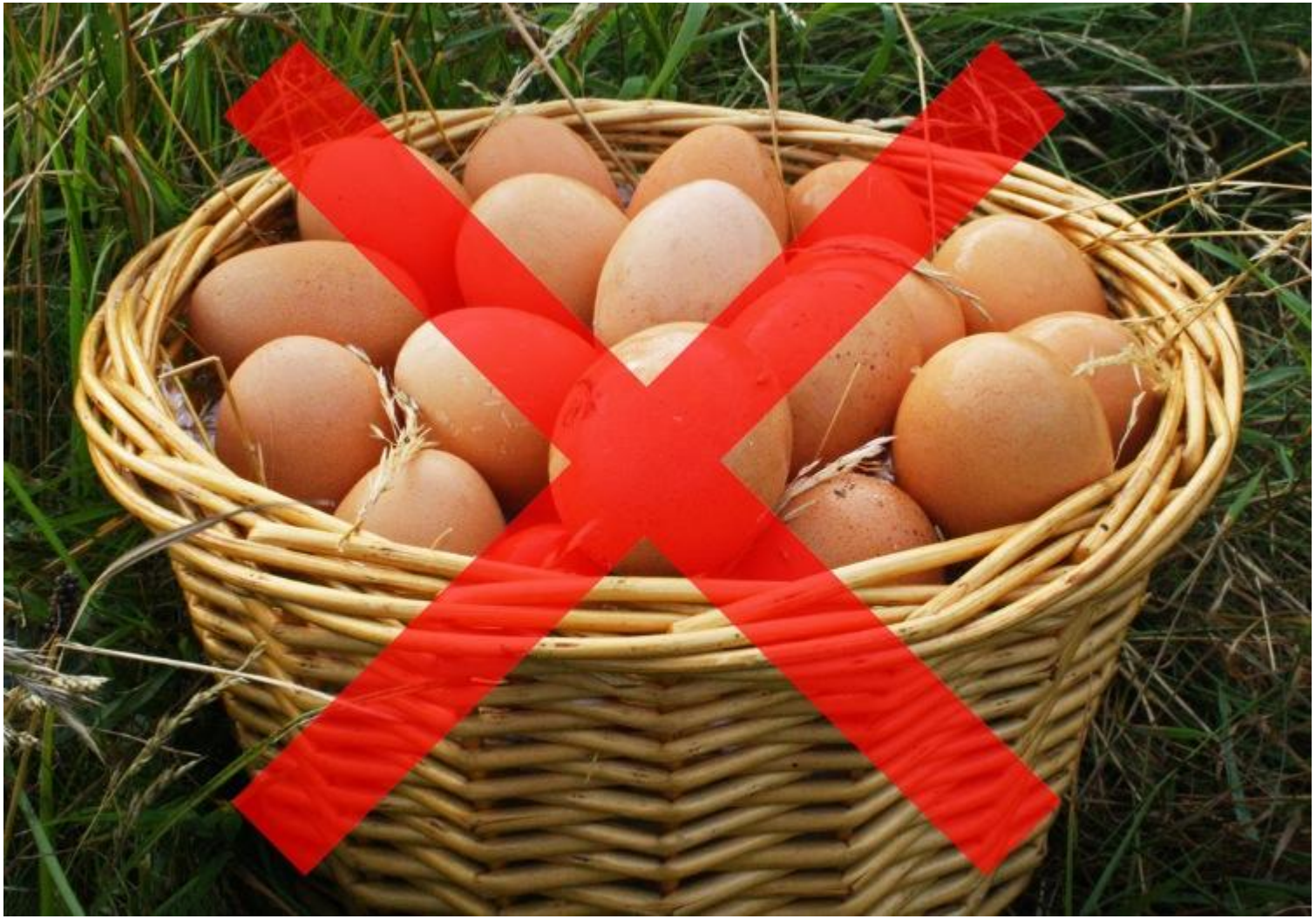
```
celery -A application worker


>>> from tasks import add_together
>>> add_together.delay(23, 42)


#loglevel

-l, --loglevel
DEBUG, INFO, WARNING, ERROR, CRITICAL, FATAL.
```

Celery. Разделение по очередям



Очереди с приоритетами

```
app.conf.task_routes = {'feed.tasks.import_feed': {'queue':
'feeds'}}

app.conf.task_routes = {'feed.tasks.*': {'queue': 'feeds'}}

app.conf.task_routes = ([
    ('feed.tasks.*', {'queue': 'feeds'}),
    ('web.tasks.*', {'queue': 'web'}),
    (re.compile(r'(video|image)\.tasks\..*'), {'queue':
'media'}),
],)
```

Пишите короткие таски

```
from utils import generate_report, send_email

@app.task()
def send_report():
    filename = generate_report()
    send_email(subject, message, attachments=[filename])
```

Установите таймауты

Установите таймауты на время выполнения:

- Через декоратор **@app.task()**, передавая **soft_time_limit**, **time_limit**
- Установив глобальный timelimit для всех тасков в очереди

Chain

```
from celery import chain

def add(a, b):
    return a + b

result = chain(add.s(2, 2), add.s(4), add.s(8))()
result.get()
```

Мониторинг выполнения тасков

```
pip install flower
flower -A application --port=5555
http://localhost:5555
```

Flower | Dashboard | Tasks | Broker | Monitor | Logout | Docs | Code

Show 10 entries Search:

Name	UUID	State	args	kwargs	Result	Received	Started	Runtime	Worker
demoapp.tasks.display_time	3d0bd4df-6db5-486d-ba2c-80f5b34de118	SUCCESS	()	{}	True	2018-01-22 17:41:57.816	2018-01-22 17:41:57.819	0.001	celery@celery-worker-7b9849b5d6-q9wbx
demoapp.tasks.display_time	0f5d833d-2007-4367-98fe-c27840045fa1	SUCCESS	()	{}	True	2018-01-22 17:41:37.816	2018-01-22 17:41:37.820	0.001	celery@celery-worker-7b9849b5d6-q9wbx
demoapp.tasks.display_time	394259c8-459c-4050-865c-c53c98ab67f2	SUCCESS	()	{}	True	2018-01-22 17:41:17.809	2018-01-22 17:41:17.811	0.001	celery@celery-worker-7b9849b5d6-q9wbx
demoapp.tasks.display_time	74348797-cf0a-4d11-92f9-acd7a61c9507	SUCCESS	()	{}	True	2018-01-22 17:40:57.804	2018-01-22 17:40:57.808	0.001	celery@celery-worker-7b9849b5d6-q9wbx
demoapp.tasks.display_time	d9c64709-20f9-4dc2-a10d-53c67b7b3648	SUCCESS	()	{}	True	2018-01-22 17:40:37.804	2018-01-22 17:40:37.806	0.001	celery@celery-worker-7b9849b5d6-q9wbx
demoapp.tasks.display_time	75596eb1-f8e5-450e-b68e-e89df9e5cbec	SUCCESS	()	{}	True	2018-01-22 17:40:17.804	2018-01-22 17:40:17.808	0.002	celery@celery-worker-7b9849b5d6-q9wbx
demoapp.tasks.display_time	c97a3397-b0e6-433b-826a-7715b4d67157	SUCCESS	()	{}	True	2018-01-22 17:39:57.803	2018-01-22 17:39:57.805	0.001	celery@celery-worker-7b9849b5d6-q9wbx
demoapp.tasks.display_time	b70df314-e50f-45c9-8fa6-bb307174d098	SUCCESS	()	{}	True	2018-01-22 17:39:37.804	2018-01-22 17:39:37.807	0.001	celery@celery-worker-7b9849b5d6-q9wbx
demoapp.tasks.display_time	e31f9dd5-5722-44c8-9d60-	SUCCESS	()	{}	True	2018-01-22	2018-01-22	0.001	celery@celery-worker-

celery beat

Особый воркер, которые умеет ставить задачи по расписанию

Типы расписаний:

- timedelta - временной интервал
- crontab - настраиваемое расписание
- solar - солнечные циклы

Запуск:

celery beat -A application

Timedelta

```
celery.conf.beat_schedule = {
    'add-every-30-seconds': {
        'task': 'tasks.add',
        'schedule': 30.0,
        'args': (16, 16)
    },
}

celery.conf.timezone = 'UTC'
```

Crontab

```
celery.conf.beat_schedule = {
    # Executes every Monday morning at 7:30 a.m.
    'add-every-monday-morning': {
        'task': 'tasks.add',
        'schedule': crontab(hour=7, minute=30, day_of_week=1),
        'args': (16, 16),
    },
}
```

Solar

```
celery.conf.beat_schedule = {
    # Executes at sunset in Melbourne
    'add-at-melbourne-sunset': {
        'task': 'tasks.add',
        'schedule': solar('sunset', -37.81753, 144.96715),
        'args': (16, 16),
    },
}

# возможные параметры: sunrise, sunset, dawn or dusk
# аргументы: solar(event, latitude, longitude)
```

Пишите письма

https://docs.djangoproject.com/en/3.0/topics/email/

```
# email server config
MAIL_SERVER = 'smtp.googlemail.com'
MAIL_PORT = 465
MAIL_USE_TLS = False
MAIL_USE_SSL = True
MAIL_USERNAME = 'your-gmail-username'
MAIL_PASSWORD = 'your-gmail-password'

# administrator list
ADMINS = ['your-gmail-username@gmail.com']
```

Пишите письма

https://docs.djangoproject.com/en/3.0/topics/email/

```
# email server config
EMAIL_HOST = 'smtp.googlemail.com'
EMAIL_PORT = 465
EMAIL_USE_TLS = False
EMAIL_USE_SSL = True
EMAIL_HOST_USER = 'your-gmail-username'
EMAIL_HOST_PASSWORD = 'your-gmail-password'

# administrator list
ADMINS = ['your-gmail-username@gmail.com']
```

Пишите письма

```
from django.core.mail import EmailMessage

msg = EmailMessage(
    "Hello",
    "testing body",
    "alena.eliz.eliz@gmail.com",
    ["ela4ka@yandex.ru"]
)

msg.send()
```

Пишите письма

```
from django.core.mail import EmailMessage
from application.celery_app import app


@app.task
def send_email(subject, sender, recipients, text):
    message = EmailMessage(
        subject,
        text,
        sender=sender,
        recipients=recipients,
    )
    message.send()


# либо воспользуйтесь готовой функцией
# from django.core.mail import send_mail
```

Домашнее задание

- Написать таск, который отправляет письмо пользователю при создании чата
- Написать периодический таск на какое-либо действие
- Использовать flower для мониторинга задач
- Использовать redis как брокер

ТЕХНОТРЕК

Спасибо за внимание!