нетология

Celery

КИРИЛЛ ТАБЕЛЬСКИЙ

КИРИЛЛ ТАБЕЛЬСКИЙ

Lightmap

План занятия

1. Что такое Celery
2. Интеграция с Flask
3. Интеграция с Django
4. Дополнительные материалы
5. Домашнее задание

Что такое Celery

Иногда серверными приложениям для обработки клиентского запроса может понадобиться много времени или ресурсов CPU/RAM . Ситуация, когда клиент ждет ответа от сервера более нескольких секунд, как правило, недопустима.

Одним из способов решения этой проблемы — создать задачу, вернуть её id клиенту, выполнять задачу в фоне и позволить клиенту проверять её выполнение, а по выполнению вернуть результат.

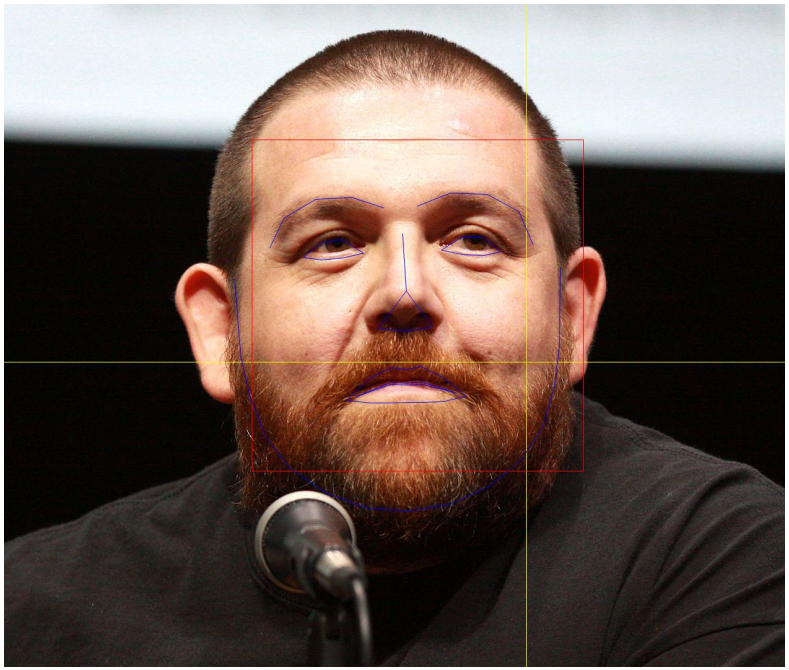
Для этого и нужен Celery.

Для чего нужен Celery?

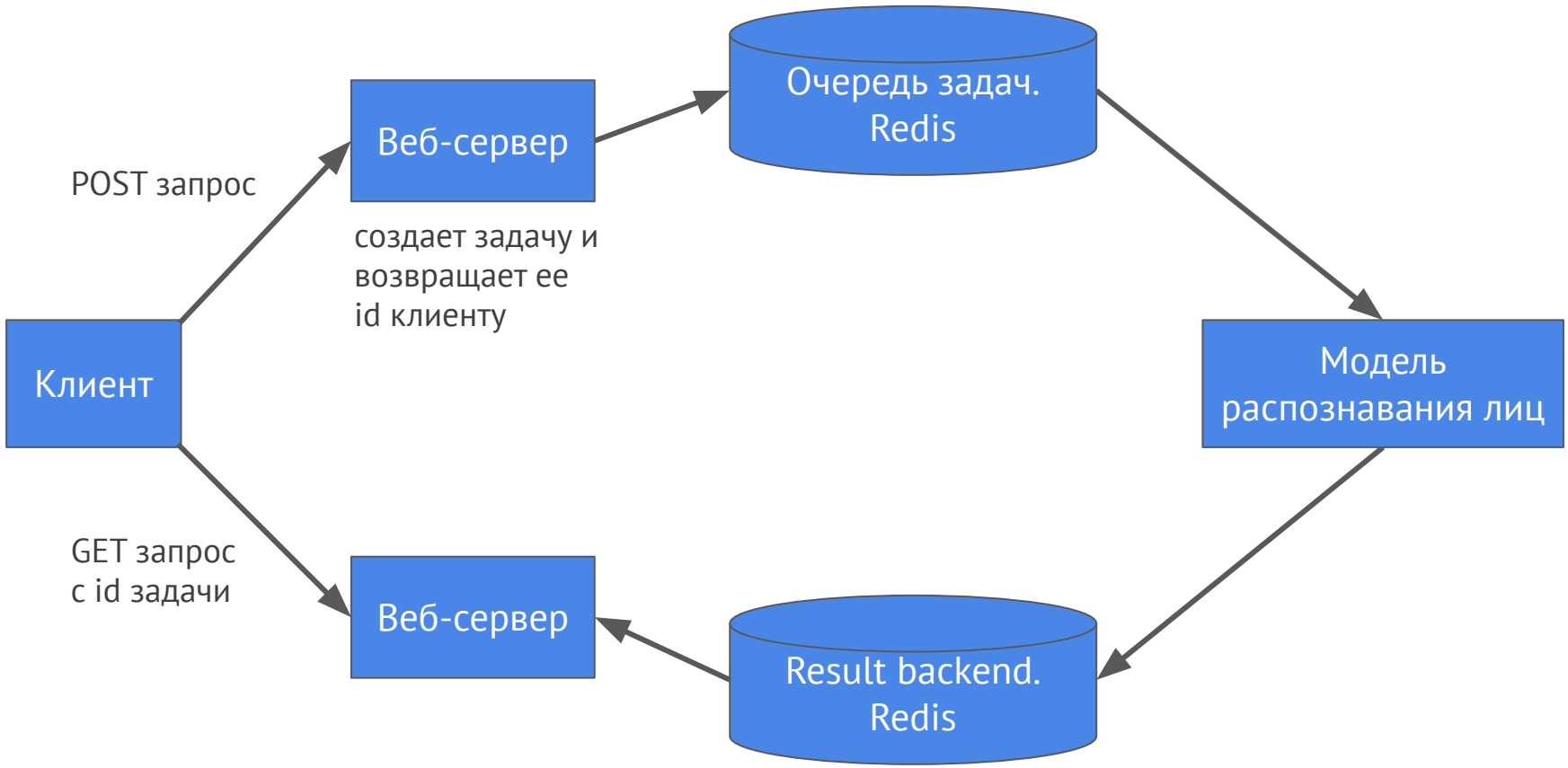
- Распараллеливание задач
- Построение очередей задач
- Асинхронное выполнение задач

Для чего нужен Celery?
Сервис распознавания лиц.

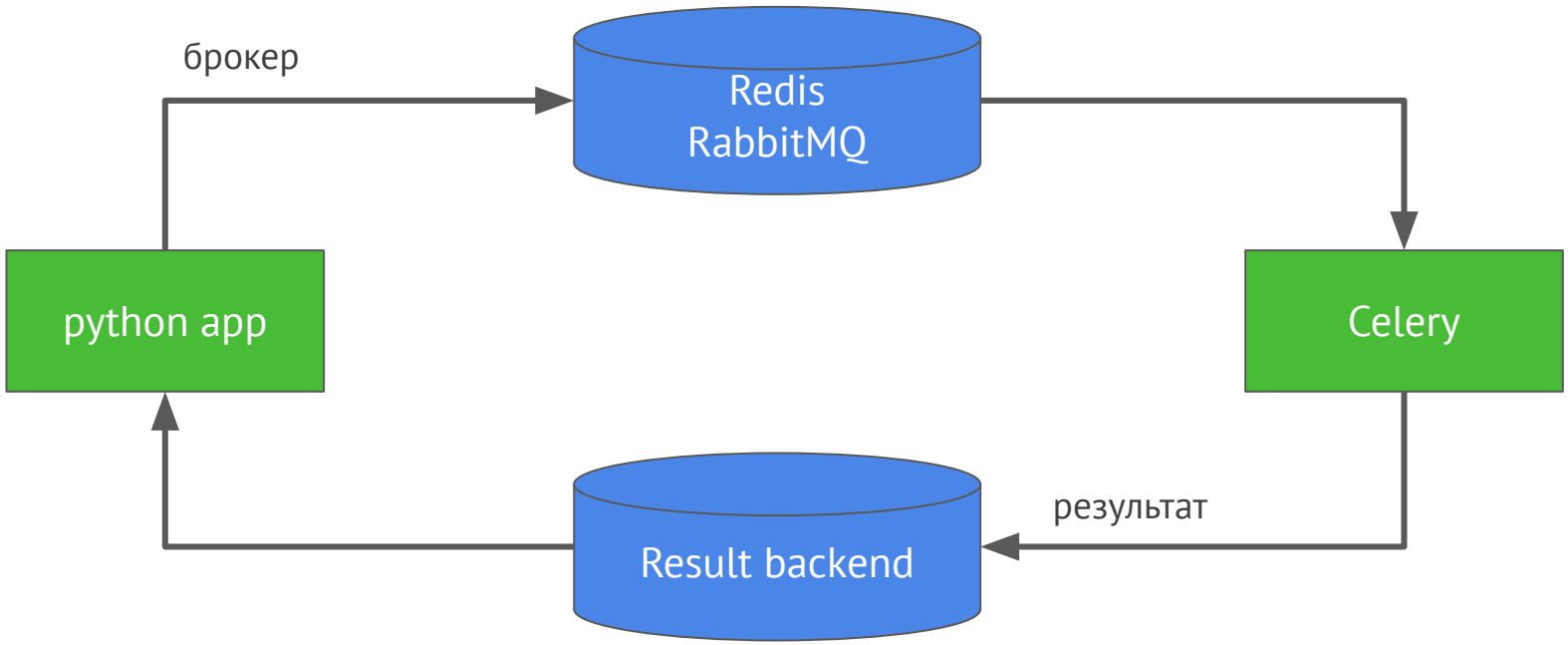
Сервис может определить: перед нами один и тот же человек?



Для чего нужен Celery?
Сервис распознавания лиц.



Общая схема работы celery



Простейшая реализация

Команда запуска Redis:

```
docker run -d -p 6379:6379 redis
```

tasks.py

```
from celery import Celery

app = Celery('tasks', broker='redis://127.0.0.1:6379/0',
backend='redis://127.0.0.1:6379/1')

@app.task
def sort_iter(data):
    return sorted(data)
```

app.py

```
from tasks import sort_iter

result = sort_iter.delay([1, 4, 2])
print(result.get())
```

```
celery -A tasks worker --loglevel=info
```

Роутинг и конфигурирование

celeryconfig.py

```
from kombu import Queue, Exchange
from tasks import sort_iter, get_sum

broker_url = 'redis://127.0.0.1:6379/0'
result_backend = 'redis://127.0.0.1:6379/1'

task_queues = (
    Queue('sort_iter_queue', Exchange('sort_iter_ex'),
routing_key='sort_iter_route'),
    Queue('get_sum_queue', Exchange('get_sum_ex'),
routing_key='get_sum_route'),
)

task_routes  = {
    'tasks.sort_iter': {'queue': 'sort_iter_queue', 'routing_key':
'sort_iter_route', 'priority': 10},
    'tasks.get_sum': {'queue': 'get_sum_queue', 'routing_key': 'get_sum_route',
'priority': 5},
}
```

```
celery worker -E -l INFO -n sort_iter -Q sort_iter_queue

celery worker -E -l INFO -n get_sum -Q get_sum_queue
```

Интеграция с Flask

celeryapp.py

```
from celery import Celery

def make_celery(app):
    celery = Celery('celeryapp')
    app.config_from_object('celeryconfig')

    class ContextTask(celery.Task):
        def __call__(self, *args, **kwargs):
            with app.app_context():
                return self.run(*args, **kwargs)

    celery.Task = ContextTask
    return celery
```

Интеграция с Flask

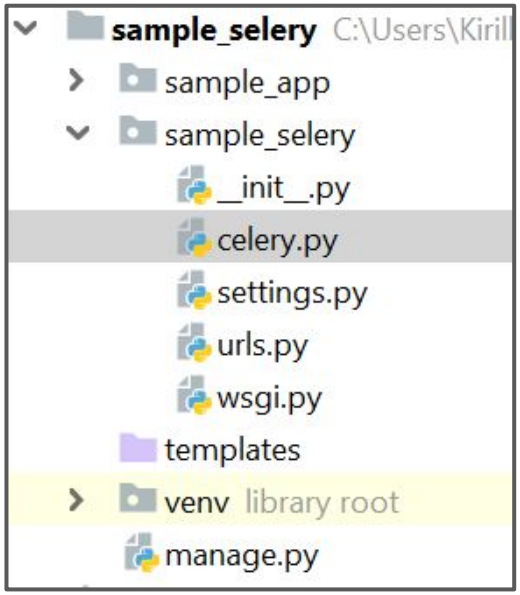
celeryconfig.py

```
from kombu import Queue, Exchange
from flaskapp import sort_iter, get_sum

broker_url = 'redis://127.0.0.1:6379/0'
result_backend = 'redis://127.0.0.1:6379/1'

task_queues = (
    Queue('default', Exchange('default'), routing_key='default'),
    Queue('sort_iter_queue', Exchange('sort_iter_ex'), routing_key='sort_iter_route'),
    Queue('get_sum_queue', Exchange('get_sum_ex'), routing_key='get_sum_route'),
)

task_routes  = {
    'flaskapp.sort_iter': {'queue': 'sort_iter_queue', 'routing_key': 'sort_iter_route', 'priority': 10},
    'flaskapp.get_sum': {'queue': 'get_sum_queue', 'routing_key': 'get_sum_route', 'priority': 5},
}
```

Интеграция с Flask

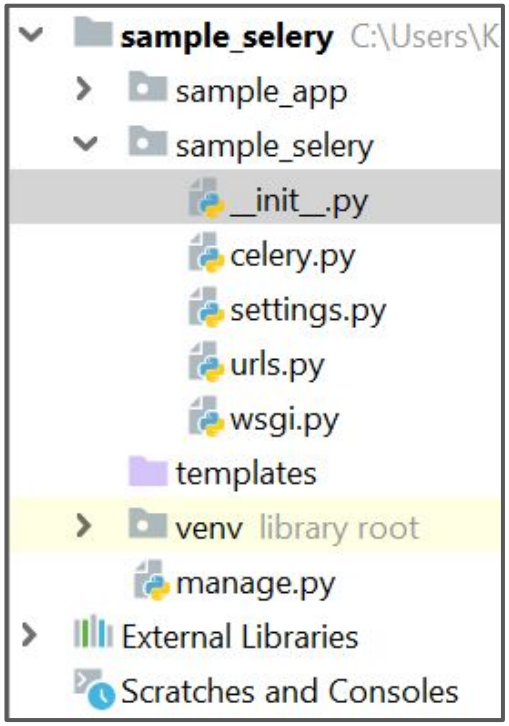
```
from flask import Flask
from celeryapp import make_celery
import json

flask_app = Flask(__name__)
flask_app.config.update(
    CELERY_BROKER_URL='redis://127.0.0.1:6379',
    CELERY_RESULT_BACKEND='redis://127.0.0.1:6379')

celery = make_celery(flask_app)

@celery.task
def sort_iter(data):
    return sorted(data)

@celery.task
def get_sum(data):
    return sum(data)

@flask_app.route('/sum/<data>')
def get_sum_view(data):
    data = [int(item) for item in data.split('!')]
    data_sum = get_sum.delay(data)
    return json.dumps(data_sum.get())

@flask_app.route('/sort/<data>')
def sort_iter_view(data):
    data = [int(item) for item in data.split('!')]
    data_sorted = sort_iter.delay(data)
    return json.dumps(data_sorted.get())
```

Интеграция с Django



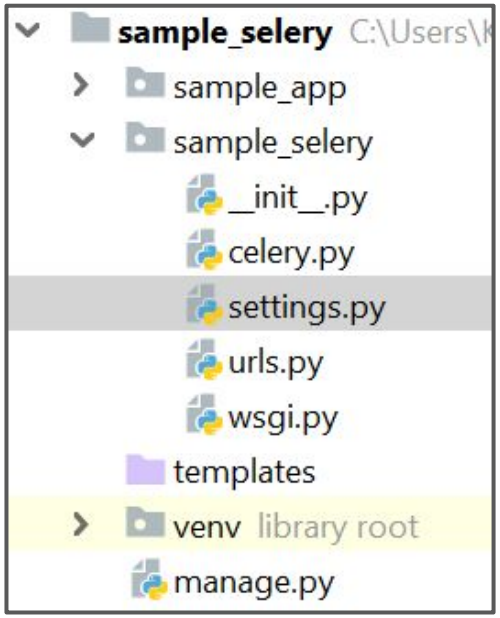

```python
from __future__ import absolute_import, unicode_literals
import os
from celery import Celery

os.environ.setdefault('DJANGO_SETTINGS_MODULE', 'proj.settings')
app = Celery('sample_selery')
app.config_from_object('django.conf:settings', namespace='CELERY')
app.autodiscover_tasks()


@app.task
def sort_iter(data):
    return sorted(data)

@app.task
def get_sum(data):
    return sum(data)
```

Интеграция с Django




```
from __future__ import absolute_import, unicode_literals
from .celery import app as celery_app

__all__ = ('celery_app',)
```

Интеграция с Django




```
INSTALLED_APPS = [
    ...
    'django_celery_results'
]

CELERY_RESULT_BACKEND = 'django-db'
CELERY_CACHE_BACKEND = 'django-cache'
os.environ.setdefault('DJANGO_SETTINGS_MODULE', 'sample_selery.settings')
```

```
pip install django-celery-results
python manage.py migrate celery_results
```

Дополнительные материалы

- http://docs.celeryproject.org/en/master/index.html/
- https://redis.io/topics/quickstart
- https://www.rabbitmq.com/install-debian.html

Домашнее задание

- Обязательного домашнего задания по лекции **нет**
- Есть **необязательное** домашнее задание

Вопросы по теме задаем в чате !

нетология

Задавайте вопросы и напишите отзыв о лекции!

КИРИЛЛ ТАБЕЛЬСКИЙ