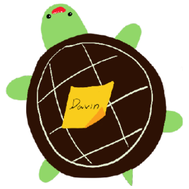
와플스튜디오 Backend Seminar[5]

변다빈 @davin111



2020.11.07.(토) 12:30 - 스프링캠프

Assignment 4

- 과제 내용과 제출 방식을 **잘 읽자** (대충 읽지 말자. 특히 과제 2는 README에 꼭 해야할 것이 있었음)
- 제출 방식을 잘 지키고(please 🙏), 과제 내용 및 제출 방식의 **취지를 생각하자**
- 과제 기간의 길이가 길다고 벼락치기 할 생각을 가급적 하지 말자 (기간의 길이가 내용에 고려됨)
- **시작하면 미리 해당 Pull Request를 생성하자** (이번엔 **자신의 repository와 rookies 모두**에!)
- Issues에서 서로 소통하자
- 즐겁게 하자

이번 Seminar 5는

- 소재를 고르기가 너무너무 어려웠어요
  - 기본적이고 쉬운 개념들은 거의 한 번씩 다 짚었고
  - 그보다 어려운 것을 제대로 하자니, 아직 앞의 내용에도 익숙해지지 못했는데 과잉이고
- 그래서 내용이 짧게 느껴지거나, 훑기만 하는 것으로 느껴질 수 있습니다
- Django 등 서버 프로그래밍 자체를 일방적으로 세미나에서 얘기하는 것은 아무 의미도 없고
  - 그것은 앞으로의 toy project 등 Wafflestudio의 프로젝트에서 직접 만들어나가야 할 부분
- 서버와 DB에 대한 소재야 무궁무진하게 엄청나게 많습니다!
  - 앞으로 좋은 서비스를 많이 개발해가며 더 고민하고 더 배워가도록 합시다!
- 남는 시간들(꽤 될 수도 있어요)은 다양한 주제에 대한 질의응답과
- 과제 4에 대한 질문을 도와드리는 시간(배포는 오프라인 상황에서 돕기가 좋습니다)으로!

Contents

오늘 이야기할 주제들

- 실제 서비스로 나아가기
  - Django Admin
  - 서버 환경 간편히 분기하기
  - select_for_update로 보는 lock 개념
  - DB 여러 개를 이용하는 서버

- 훑어보기
  - 비동기, ASGI, Django 3
  - Celery, task queue
  - Cron
  - Spring, Node.JS

https://docs.djangoproject.com/en/3.1/ref/contrib/admin/
https://developer.mozilla.org/ko/docs/Learn/Server-side/Django/Admin_site

Django Admin

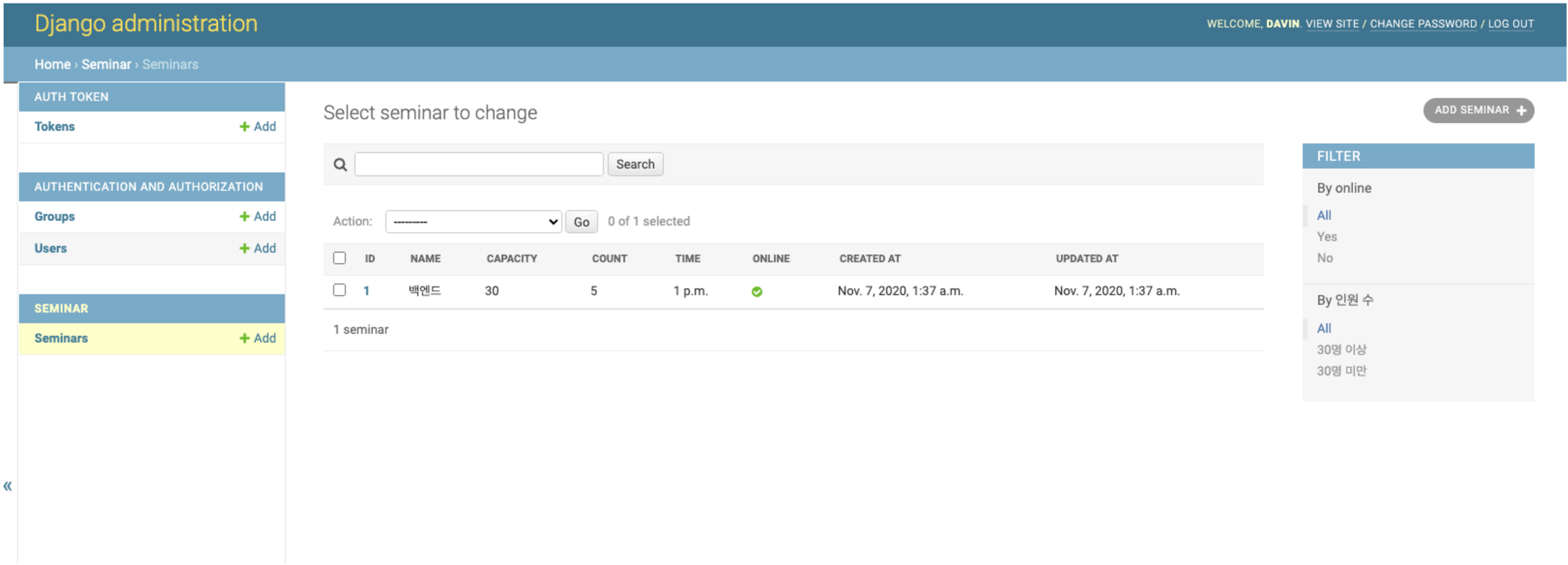
어쩌면 Django를 택하는 큰 이유 중 하나

- 서비스를 만들어가는 모든 사람이 개발자는 아니다
- 개발자 외의 누군가가 ‘제한된’ 상황에서 ‘편리하게’ DB의 데이터를 조회하고 변경하려면?
  - SQL을 공부하게 한다
  - API를 개발하고 이를 이용하는 프론트(staff 페이지)를 개발한다
  - Django Admin을 이용한다
  - …

Django Admin

어쩌면 Django를 택하는 큰 이유 중 하나

- 매우 빠른 개발 생산성을 갖고 staff 페이지를 만들 수 있음
- 직접 만들고 살펴보기!



https://medium.com/@bdv111/django-admin의-검색을-위한-search-fields가-db에-일으킬-수-있는-성능-문제-3f816aff293e

Django Admin

어쩌면 Django를 택하는 큰 이유 중 하나

- 작은 회사에서 사내용 간단한 웹페이지를 만드는 것은 서버 개발자의 몫이 되기도 함
- admin 페이지를 만드는 것은 다소 지루하고 단순한 작업처럼 여겨지나, 잘 하려면 어려움
  - DB에 서비스 로직상 이상한 상태의 데이터가 만들어지지 않도록 잘 설계해야 함
  - 권한이 없는 사람이 함부로 특정 기능을 이용할 수 없도록 보안을 유지해야 함
  - admin 페이지가 실제 서비스의 성능에 영향을 주지 않도록 주의해야 함

https://github.com/wafflestudio/rookies/issues/255
https://docs.djangoproject.com/en/3.1/topics/settings/#designating-the-settings

서버 환경 간편히 분기하기

여러 방법 중 하나를 소개합니다

- 우리가 지금 이용하고 있는 DB들을 생각해보면: 나의 local DB, AWS RDS의 DB
- local에서 개발할 때, 두 DB 각각을 연결해보고 싶을 수 있음
  - 매번 settings.py의 DATABASES를 수정할 것인가?
  - 환경 변수를 이용하자!
  - DJANGO_SETTINGS_MODULE에 대해서도 참고하기
- 클라우드 컴퓨팅 서비스를 이용할 때도 dev, prod 환경 등으로 나눠 운영하는 경우가 많음

https://docs.djangoproject.com/en/3.1/ref/models/querysets/#select-for-update
https://medium.com/@chrisjune_13837/db-lock-락이란-무엇인가-d908296d0279

select_for_update()로 보는 lock 개념

lock은 컴퓨터 제반에서 중요한 개념입니다

- 내가 이 row들 건드리는 동안 다른 놈은 건드리지 마!
  - 이러한 각각의 주체가 transaction
  - 실제 SQL로 SELECT … FOR UPDATE 가 됨
  - 성능(concurrency)을 희생하여 일관성(consistency)의 원칙을 지킨다 (바로 뒤에서 ACID 복습)
  - 남발하면 큰일 남, 그리고 lock을 다룰 때는 deadlock을 주의해야 함

- DB table에 대한 의도치 않은/불가피한 lock은 장애를 유발하기 쉽다
  - write/read가 복잡하게 꼬여있는 로직, 잘못 만든 query, row가 많은 table에 대한 migration 등

ACID (revisited)
DB transaction의 원칙

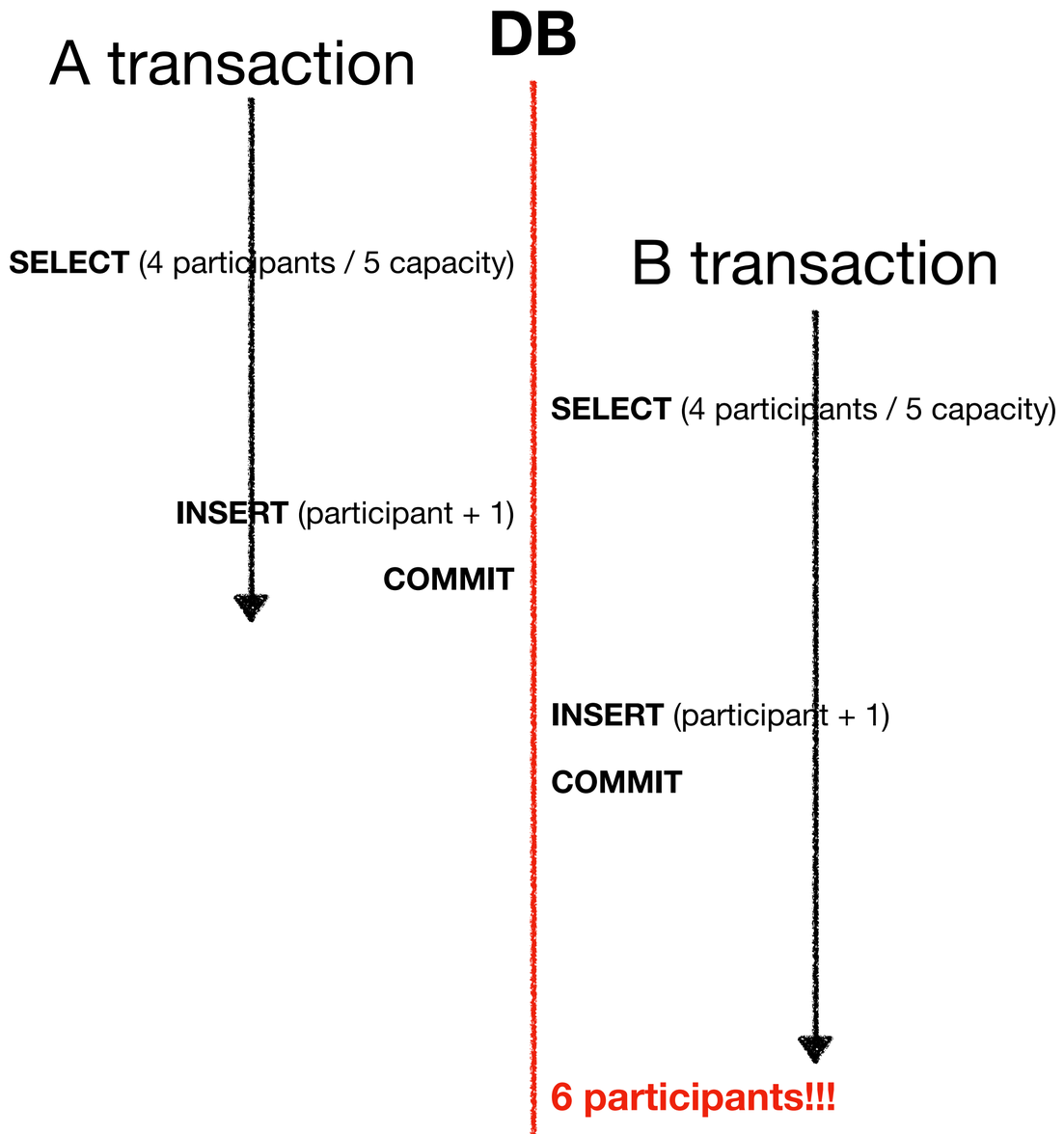
- Atomicity (원자성)
  - 다 되거나 다 안 되거나, 둘 중에 하나여야 함
- Consistency (일관성)
  - 미리 정해둔 규칙에 맞게 데이터를 저장, 변경해야 함
- Isolation (고립성)
  - 복수 개의 transaction이 실행될 때 서로 연산이 끼어들면 안 되며, 중간 과정이 이용되어도 안 됨
- Durability (영구성)
  - 일단 commit했으면 어떠한 문제가 발생해도 영구적으로 반영된 상태여야 함

https://github.com/wafflestudio/rookies/issues/223

동시성을 고려해 transaction.atomic 사용하기

Seminar 4에서의 내용까지 빠르게 짚자면

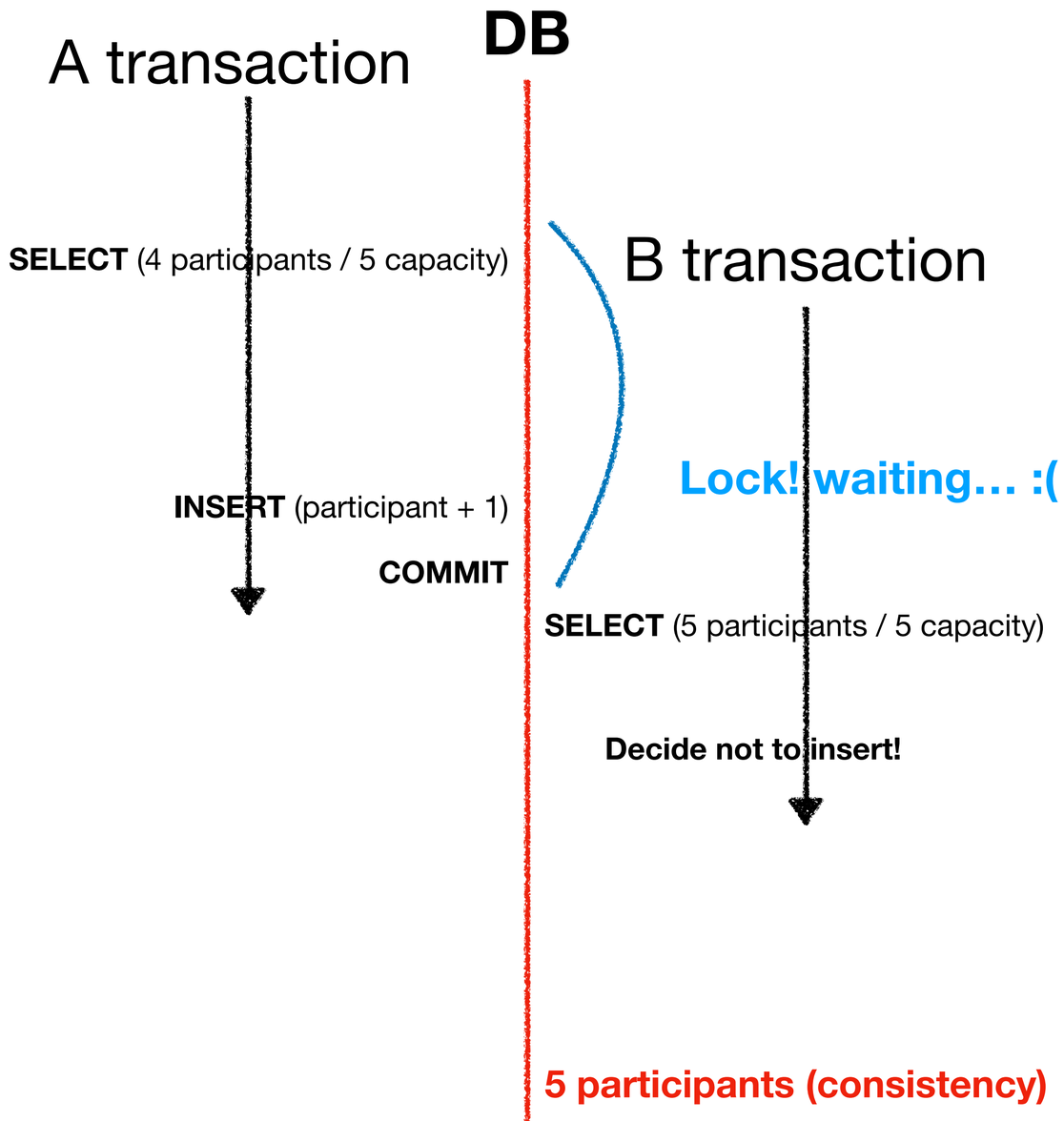
- POST /api/v1/seminar/user/ 를 이용해 participant로서 Seminar 참여하기
- 짧은 시간 내에 여러 명이 capacity가 거의 다 찬 Seminar에 참여하면 어떡하나?
- get과 create가 별개의 transaction에 속하면 lock이고 뭐고 의미가 없음
- 일단 하나의 API 내에서의 SELECT와 INSERT query를 하나의 transaction 내에서 수행되도록 묶는다



동시성을 고려해 lock 사용하기

Seminar 4에서의 내용에 이어서!

- 그런데 그림에서 볼 수 있듯 해결이 안 됨
- A transaction이 SELECT하고 INSERT할 때까지 B transaction이 아예 건드리면 안 됨
- 그러니까 SELECT … FOR UPDATE를 이용해 exclusive lock(read도 못함)을 건다
- consistency는 지켜지지만 성능을 희생하고 있음을 인지해야 함



https://docs.djangoproject.com/en/3.1/ref/settings/#databases
https://docs.djangoproject.com/en/3.1/ref/models/querysets/#using

DB 여러 개를 이용하는 서버

한 서버가 동시에 DB 여러개를 쓴다구요?

- 서비스가 복잡해지면, 원해서든 부득이하게든 하나의 서버가 여러 개의 DB를 보아야 함
- dict 형태에 ‘default’로 되어있는 것이 해당 서버의 기본 DB이며, 그 외에 이름을 정해서 추가하면 됨

```
94  DATABASES = {
95      'default': {
96          'ENGINE': 'django.db.backends.mysql',
97          'HOST': 'localhost',
98          'PORT': 3306,
99          'NAME': 'waffle_backend_assignment_2',
100         'USER': 'waffle-backend',
101         'PASSWORD': 'seminar',
102     }
103 }
```

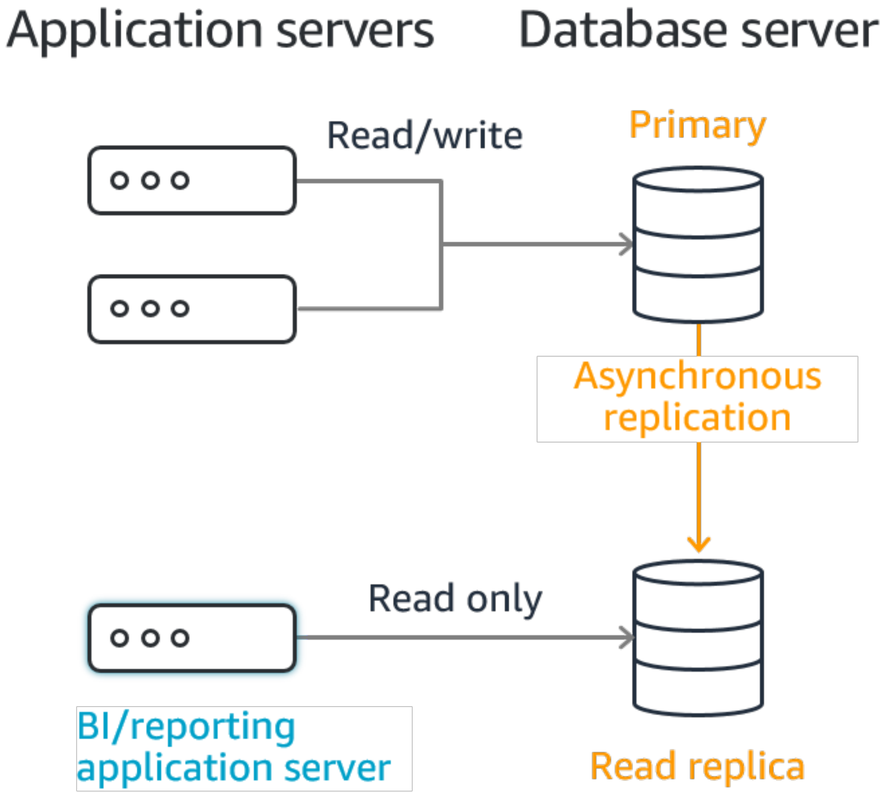
- 당연히 다른 DB끼리는, 설령 데이터끼리 관계성이 있어도 join 등 하나의 query로 가져올 수 없어서 서버 프로그래밍의 고민이 더 높아짐

https://aws.amazon.com/ko/rds/features/read-replicas/

DB 여러 개를 이용하는 서버

데이터가 같은 DB를 여러 개 연결하기도 한다

- 작은 앱 서버는 하나의 DB에 대해 read/write를 다 하는 것이 자연스러움
- 그런데 그 DB가 터지면?
- DB 컴퓨터가 아무리 좋아도 하나만으로는
  감당할 수 있는 connection, query의 한계가 있음



- Query 유형 중 가장 많은 것은 무엇일까?
- SELECT를 위해 Primary / Read relplica를 구분함
- write할 때 이용하는 DB를 기본으로 두고
  Read replica를 한 개 이상 별도로 두는 식
- 당연히 replica lag(write된 것이 늦게 반영)가 발생 가능

https://asgi.readthedocs.io/en/latest/
https://docs.djangoproject.com/en/3.1/releases/3.0/

지금부터의 주제들은 더 간단히

이런 게 있구나~ 정도만!

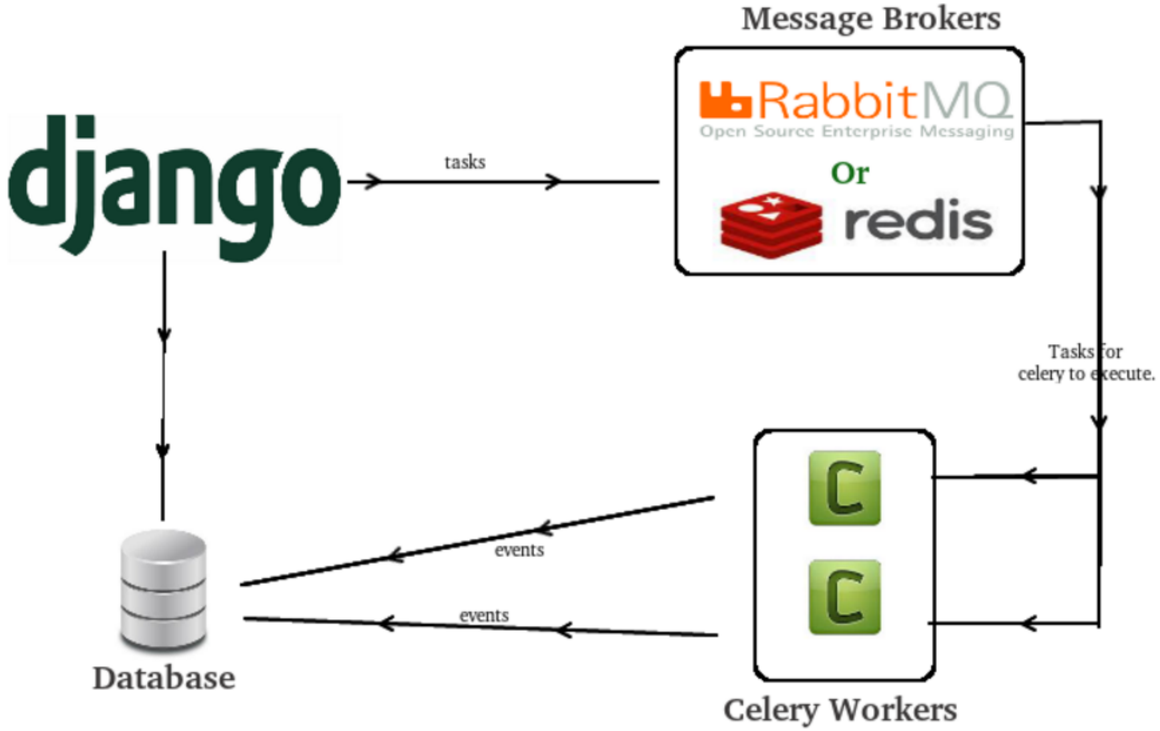
- 동기적으로 운영되는 서버의 단점은 무엇일까? 동기/비동기가 뭐지?
- 서브웨이 매장의 점원 분들이 일하시는 방식을 생각해 봅시다
  - 한 명의 점원(process)이 한 명의 손님(request)를 처음부터 끝까지 처리하나요?
  - 어떻게 하는 것이 더 빠를까요?

https://docs.celeryproject.org/en/stable/
https://beomi.github.io/2016/11/04/ebb288ec97ad-ec9ea5eab3a0djangoec9980-ed95a8eabb98ed9598eb8a94-celery-ecb2abeab1b8ec9d8c/
https://spoqa.github.io/2012/05/29/distribute-task-with-celery.html

Celery

Distributed Task Queue

- 반드시 response에 포함되지 않아도 되고, 늦게라도 처리만 하면 되는 경우
- 뒤에서 언급할 정기적인 cron 등을 위해서도 사용됨
- Celery 자체만으로는 사용할 수 없고
  message broker로 사용할 무언가가 필요함

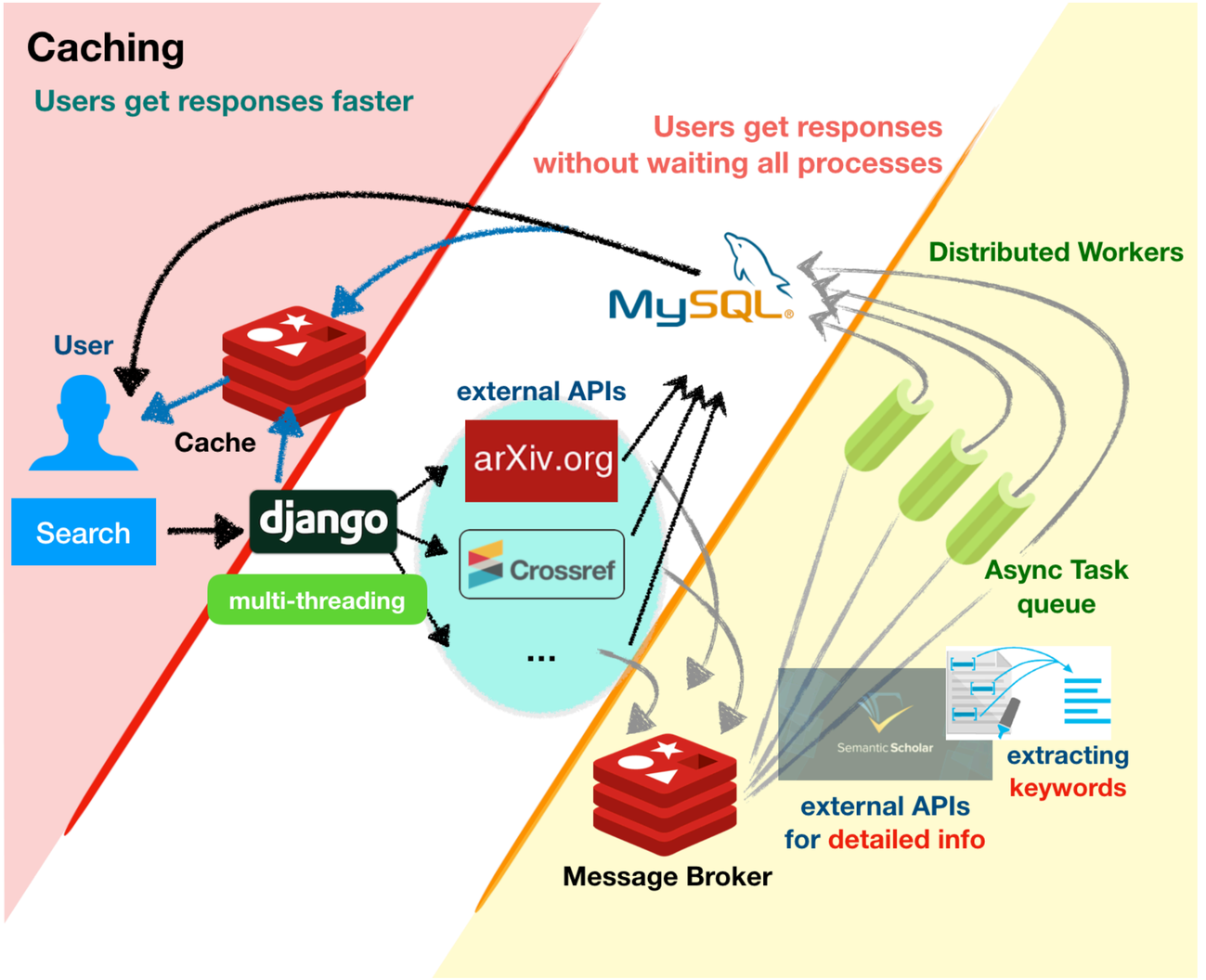


https://github.com/swsnu/swpp2019-team3/wiki/System-Architecture

Celery

Distributed Task Queue

- 과거 PapersFeed 프로젝트에서의 architecture 이미지
- Celery는 비동기/분산의 장점을 활용 가능



https://ko.wikipedia.org/wiki/Cron
https://jhnyang.tistory.com/68

Cron

Job Scheduler

- Cron 자체는 unix 계열 운영체제의 job scheduler를 말함
- 어떤 서비스든 scheduling 하여 수행되는 일이 필요해짐 (특히 무거운 job들)
  - 이러한 것들을 trigger 시키는 방법으로서 cron의 개념, 또는 그 문법이 자주 이용됨

```
# ┌──────────────── min (0 - 59)
# │ ┌────────────── hour (0 - 23)
# │ │ ┌──────────── day of month (1 - 31)
# │ │ │ ┌────────── month (1 - 12)
# │ │ │ │ ┌──────── day of week (0 - 6) (0 to 6 are Sunday to Saturday, or use names; 7 is Sunday, the same as 0)
# │ │ │ │ │
# │ │ │ │ │
# * * * * *  command to execute
```

  - 예를 들어 0 20 * * * 은 매일 오후 8시에 특정 프로그램을 수행시키겠다는 것
- trigger 시킬 방법, 그리고 그것을 수행할 방법이 고려되면 간단함

Other server frameworks

Spring Boot(JPA), Node.js 등

- Python으로 서버를 구축하는 데에 이용할 수 있는 framework들도 더 있습니다
  - sanic, alembic, sqlalchemy 등

- Python, Django와 장단점이 다른 여러 framework들도 익히는 것이 좋습니다!
  - Spring Boot(JPA), Node.js 등
  - 저도 공부하는 중!

Any Questions?

또는 세미나장이 하고 싶은 말, 못다한 말

질문 많이 해주세요!

Toy project, 그리고 이후 Wafflestudio 활동에서
재밌는 것들을 함께 많이 만들어봅시다! 🦄🌏🚀