PNU Mini Bootcamp 백엔드&클라우드 과정

6일차 - Redis와 캐싱 전략

송준우

2025년 2월 10일

멋쟁이사자처럼

목차

1. Redis와 캐싱 전략

1.1.1 Redis

Redis는 Key-Value형태로 데이터를 메모리에 저장하는 In-Memory 데이터베이스입니다.

- 빠른 읽기/쓰기 속도
- Key-Value 저장 구조
- 데이터 캐싱, 세션 관리, 메세지 큐로도 활용 가능
- TTL(Time To Live)
- Pub/Sub 이벤트
- 디스크 저장 지원(스냅샷, 로그기반 복구)

1.1.1 Redis

Redis 주요 활용 사례

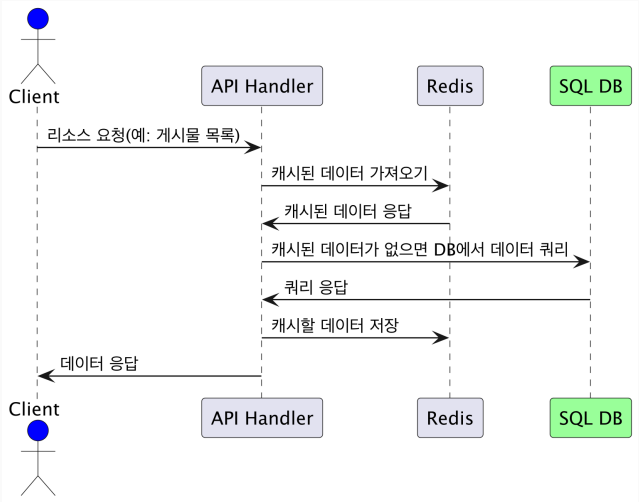
- API 응답 캐싱
- 웹 세션 관리
- 검색어, 추천 키워드 자동완성
- 실시간 랭킹

1.1.1 Redis

주의사항

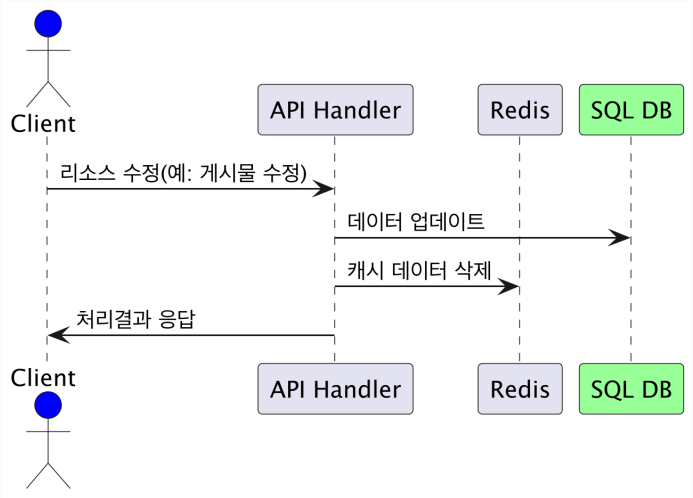
- 저장공간: 대용량, 대량 데이터 저장 불가
- 데이터 유실 대비: 서버 중단시 데이터 유실
- SQL 사용불가: JOIN등 복잡한 쿼리 사용 불가
- 데이터 영구 저장 불가

1.1.2 API 캐싱



캐싱 및 캐시된 데이터 가져오기

1.1.2 API 캐싱



캐시데이터 갱신

1.2.1 Redis 설치

Windows

- 바이너리 설치(버전 3): https://github.com/microsoftarchive/redis/releases/tag/win-3.0.504
- WSL2를 이용한 설치:
  https://redis.io/docs/latest/operate/oss_and_stack/install/install-redis/install-redis-on-windows/

macOS

```
brew install redis
```

Ubuntu Linux

```
sudo apt install redis redis-tools -y
```

* 또는 Docker

1.2.2 Redis 기초

Redis CLI 클라이언트 실행

```
redis-cli
```

Redis CLI도구를 이용해 기본적인 사용법을 학습해봅시다.

1.2.2 Redis 기초

Redis 주요 데이터 타입

- String
- List
- Set
- Sorted set
- Hashes

1.2.2 Redis 기초

- KEYS *: 모든 키 목록
- DBSIZE: 현재 사용중인 DB크기
- SET key value [EX expirySeconds] [NX—XX]: 키=값 추가, 만료시간 초단위로 설정, NX=키가 없으면 추가
- GET key: key에 해당하는 값 가져오기
- DEL key key ...: 주어진 키들 삭제하기
- EXISTS key: key에 해당하는 값이 있는지 검사
- EXPIRE key seconds: key에 만료시간 초단위로 설정
- PERSIST key: key에 만료시간 제거
- TTL key: key에 남은 시간 조회
- INCR key: key의 값 1증가
- DECR key: key의 값 1감소

List 형식

- LPUSH key value: 리스트의 앞쪽에 값 추가
- RPUSH key value: 리스트의 뒤쪽에 값 추가
- LPOP key [count]: 리스트의 왼쪽 값 꺼내기
- RPOP key [count]: 리스트의 오른쪽값 꺼내기
- LRANGE key start stop: 리스트의 값 조회

```
LPUSH user "Linux" "Windows" "macOS"
LRANGE user 0 -1
LPOP user
LPOP user 2
```

Set 형식

- SADD key member member ..: key에 member들 추가
- SMEMBERS key: key의 멤버들 조회
- SCARD key: key의 멤버 수
- SISMEMBER key member: member가 key에 속해있는지
- SREM key member member...: key에서 member들 삭제
- SPOP key [count]: key의 멤머 count개를 무작위로 꺼내기
- SRANDMEMBER key [count]: key의 멤버 count개를 무작위로 가져오기
- SINTER key key...: key들간의 교집합 가져오기
- SUNION key key...: key들간의 합집합 가져오기
- SDIFF key key...: key들간의 차집합 가져오기

Sorted set 형식

- ZADD key score member: key에 점수와 항목 추가
- ZINCRBY key increment member: key의 member의 score를 increment만큼 증가
- ZRANGE key start stop [WITHSCORES] [REV]: key의 값 조회, REV=내림차순
- ZRANK key member [REV]: key에서 member의 순위 조회, REV=내림차순
- ZSCORE key member: key에서 member의 점수 조회
- ZCOUNT key min max: key에서 score가 min∼max 사이인 멤버 조회
- ZREM key member member..: key에서 멤버 삭제

Hashes 형식

- HSET key field1 value1 [field2 value2],...
- HGET key field: key 해시의 field에 해당하는 값 가져오기
- HDEL key field: key 해시의 field 삭제
- HEXISTS key field: key 해시에 field가 있는지 검사
- HGETALL key: key 해시의 모든 필드와 값 가져오기
- HKEYS key: key 해시의 모든 필드 가져오기
- HLEN key: key 해시의 필드 수 가져오기

1.3.1 투표 API 예제

FastAPI와 Redis를 연동하여 투표시스템을 만들어봅시다.
새로운 FastAPI 프로젝트를 생성합니다.

```
mkdir VoteAPI
cd VoteAPI
python3 -m venv .venv
source .venv/bin/activate
(.venv) pip3 install "FastAPI[standard]"
(.venv) pip3 install aioredis
```

* aioredis 레퍼런스: https://aioredis.readthedocs.io/en/v1.3.0/mixins.html

1.3.1 투표 API

VoteAPI는 투표하기와 현재까지의 투표기록을 조회하는 기능을 제공합니다.

- GET /votes: 투표 대상 목록 조회
- PUT /votes/member_id: member_id에게 투표하기
- GET /score: 현재까지의 점수 조회

main.py 파일을 생성하고 아래와 같이 기본 코드를 작성합니다.

```
from fastapi import FastAPI, Depends, Path
from typing import Annotated
import aioredis

app = FastAPI()
REDIS_URL = "redis://localhost"
async def get_redis():
    return await aioredis.from_url(REDIS_URL, decode_responses=True)
KEY_VOTES = "votes"
MEMBERS = ['Linux', 'macOS', 'Windows']
```

1.3.1 투표 API

아래와 같이 코드를 작성하고 http://localhost:8000/docs/에 접속하여 API를 테스트해봅시다.

투표대상 목록 조회

```
@app.get('/votes')
def votes() -> list[str]:
    return MEMBERS
```

1.3.1 투표 API

투표하기

```
@app.put('/votes/{member_id}')
async def vote_to(member_id: Annotated[int, Path(ge=0, lt=len(MEMBERS))],
                  redis=Depends(get_redis)):
    await redis.zincrby(KEY_VOTES, 1, MEMBERS[member_id])
    return {}
```

Redis Cli에서 아래 명령어를 입력하여 테스트한 투표수가 반영되었는지 확인해보세요.

```
zrange votes 0 -1 withscores rev
```

1.3.1 투표 API

Redis의 ZRANGE 명령을 이용해 득표수를 내림차순으로 가져옵니다.

투표현황 조회

```
@app.get('/score')
async def score(redis=Depends(get_redis)):
    scores = await redis.zrevrange(KEY_VOTES, 0, -1, True)
    return scores
```

1.3.1 투표 API

실습

aioredis Reference를 참고하여 아래 기능들을 구현해보세요.

1. 특정 Member의 현재 득표수를 조회하는 API를 구현해보세요.
2. 투표하기 API에 사용자의 전화번호를 입력받고 해당번호로 중복 투표가 불가능하도록 수정해보세요.

* aioredis 레퍼런스: https://aioredis.readthedocs.io/en/v1.3.0/mixins.html

1.4 개인프로젝트 캐싱 적용

진행중인 프로젝트에 캐싱을 적용해봅시다.