链滴

RedisTemplate 随意切换多个 db 库 不影响其他线程

作者：724555508

原文链接：https://ld246.com/article/1594863004207

来源网站：链滴

问题描述

原切换 db 的方式会使全局的 redisTemplate 被修改掉，并发量稍微高一点，就会影响到其他线程查 Redis 数据

```
//原切换db的方式（错误的）
public RedisTemplate<Serializable, Object> initRedis(Integer indexDb){
        LettuceConnectionFactory jedisConnectionFactory = (LettuceConnectionFactory) redisT
mplate.getConnectionFactory();
        jedisConnectionFactory.setDatabase(indexDb);
        redisTemplate.setConnectionFactory(jedisConnectionFactory);
        jedisConnectionFactory.resetConnection();
        return redisTemplate;
}
```

解决方案，为每一个 db 创建一个 redisTemplate

```
import java.io.Serializable;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

import javax.annotation.PostConstruct;

import org.springframework.beans.factory.annotation.Value;
import org.springframework.data.redis.connection.RedisConnectionFactory;
import org.springframework.data.redis.connection.RedisPassword;
```

```
import org.springframework.data.redis.connection.RedisStandaloneConfiguration;
import org.springframework.data.redis.connection.jedis.JedisClientConfiguration;
import org.springframework.data.redis.connection.jedis.JedisConnectionFactory;
import org.springframework.data.redis.core.RedisTemplate;
import org.springframework.data.redis.serializer.GenericJackson2JsonRedisSerializer;
import org.springframework.data.redis.serializer.StringRedisSerializer;
import org.springframework.stereotype.Component;

import lombok.extern.slf4j.Slf4j;
import redis.clients.jedis.JedisPoolConfig;

@Component
@Slf4j
public class RedisConfig {

    @Value("${redis.host}")
    private String hostName;
    @Value("${redis.port}")
    private int port;
    @Value("${redis.password}")
    private String passWord;
    @Value("${redis.maxIdle}")
    private int maxIdl;
    @Value("${redis.minIdle}")
    private int minIdl;
    @Value("${redis.timeout}")
    private int timeout;

    private int defaultDb;

    @Value("${redis.dbs}")
    private List<Integer> dbs;


    public static Map<Integer, RedisTemplate<Serializable, Object>> redisTemplateMap = new
HashMap<>();

    @PostConstruct
    public void initRedisTemp() throws Exception {
        log.info("###### START 初始化 Redis 连接池 START ######");
        defaultDb = dbs.get(0);
        for (Integer db : dbs) {
            log.info("###### 正在加载Redis-db-" + db+ " ######");
            redisTemplateMap.put(db, redisTemplateObject(db));
        }
        log.info("###### END 初始化 Redis 连接池 END ######");
    }

    public RedisTemplate<Serializable, Object> redisTemplateObject(Integer dbIndex) throws
xception {
        RedisTemplate<Serializable, Object> redisTemplateObject = new RedisTemplate<Seriali
able, Object>();
        redisTemplateObject.setConnectionFactory(redisConnectionFactory(jedisPoolConfig(), db
ndex));
```

```
        setSerializer(redisTemplateObject);
        redisTemplateObject.afterPropertiesSet();
        return redisTemplateObject;
    }

    /**
     * 连接池配置信息
     *
     * @return
     */
    public JedisPoolConfig jedisPoolConfig() {
        JedisPoolConfig poolConfig = new JedisPoolConfig();
        // 最大连接数
        poolConfig.setMaxIdle(maxIdl);
        // 最小空闲连接数
        poolConfig.setMinIdle(minIdl);
        poolConfig.setTestOnBorrow(true);
        poolConfig.setTestOnReturn(true);
        poolConfig.setTestWhileIdle(true);
        poolConfig.setNumTestsPerEvictionRun(10);
        poolConfig.setTimeBetweenEvictionRunsMillis(60000);
        // 当池内没有可用的连接时，最大等待时间
        poolConfig.setMaxWaitMillis(10000);
        // ------其他属性根据需要自行添加-------------
        return poolConfig;
    }

    /**
     * jedis连接工厂
     *
     * @param jedisPoolConfig
     * @return
     */
    public RedisConnectionFactory redisConnectionFactory(JedisPoolConfig jedisPoolConfig, in
 db) {
        // 单机版jedis
        RedisStandaloneConfiguration redisStandaloneConfiguration = new RedisStandaloneCo
figuration();
        // 设置redis服务器的host或者ip地址
        redisStandaloneConfiguration.setHostName(hostName);
        // 设置默认使用的数据库
        redisStandaloneConfiguration.setDatabase(db);
        // 设置密码
        redisStandaloneConfiguration.setPassword(RedisPassword.of(passWord));
        // 设置redis的服务的端口号
        redisStandaloneConfiguration.setPort(port);

        // 获得默认的连接池构造器(怎么设计的，为什么不抽象出单独类，供用户使用呢)
        JedisClientConfiguration.JedisPoolingClientConfigurationBuilder jpcb = (JedisClientConfi
uration.JedisPoolingClientConfigurationBuilder) JedisClientConfiguration
                .builder();
        // 指定jedisPoolConifig来修改默认的连接池构造器（真麻烦，滥用设计模式！）
        jpcb.poolConfig(jedisPoolConfig);
        // 通过构造器来构造jedis客户端配置
```

```
        JedisClientConfiguration jedisClientConfiguration = jpcb.build();
        // 单机配置 + 客户端配置 = jedis连接工厂
        return new JedisConnectionFactory(redisStandaloneConfiguration, jedisClientConfigurati
n);
    }

    private void setSerializer(RedisTemplate<Serializable, Object> template) {
        template.setKeySerializer(new StringRedisSerializer());
        template.setValueSerializer(new GenericJackson2JsonRedisSerializer());
    }


    public RedisTemplate<Serializable, Object> getRedisTemplateByDb(int db){
            return redisTemplateMap.get(db);
        }

    public RedisTemplate<Serializable, Object> getRedisTemplate(){
            return redisTemplateMap.get(defaultDb);
        }
}
```

application.yml

```
redis:
    host: 192.168.100.77
    port: 6379
    password: 123456      # 密码（默认为空）
    maxIdle: 10
    minIdle: 5
    timeout: 1000
    dbs: 0,1      # 可以指定多个，用“,”隔开，写在第一个的为defaultDB
```

使用方法

```
@Resource RedisConfig redisConfig;

//无参数时默认查询defaultDB  也就是配置文件中写在第一个的db
redisConfig.getRedisTemplate().opsForValue().....

//可以指定具体查询哪个DB
redisConfig.getRedisTemplate(1).opsForValue().....
```