Atitit RedisTemplate的使用

单独使用。。

切换数据库

日志记录

package com.wz.common.redis;

import java.io.IOException;

import java.io.InputStream;

import java.util.HashMap;

import java.util.LinkedHashMap;

import java.util.List;

import java.util.Map;

import java.util.Properties;

import java.util.ResourceBundle;

import java.util.Set;

import java.util.concurrent.TimeUnit;

import org.apache.commons.lang3.StringUtils;

import org.apache.commons.logging.Log;

import org.apache.commons.logging.LogFactory;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.data.redis.connection.RedisStandaloneConfiguration;

import org.springframework.data.redis.connection.jedis.JedisClientConfiguration;

import org.springframework.data.redis.connection.jedis.JedisClientConfiguration.DefaultJedisClientConfigurationBuilder;

import org.springframework.data.redis.connection.jedis.JedisConnectionFactory;

import org.springframework.data.redis.connection.lettuce.LettuceConnectionFactory;

import org.springframework.data.redis.core.RedisTemplate;

import org.springframework.data.redis.core.StringRedisTemplate;

import org.springframework.data.redis.serializer.RedisSerializer;

import org.springframework.data.redis.serializer.StringRedisSerializer;

import org.springframework.stereotype.Component;

import org.springframework.util.CollectionUtils;

import com.alibaba.fastjson.JSON;

import com.alibaba.fastjson.JSONObject;

import com.google.common.collect.Maps;

import lombok.val;

@Component

public final class RedisUtils {

@Autowired

public RedisTemplate<String, Object> redisTemplate;

/\*\*

\* 对key上锁

\*

\* @param key

\* @return

\*/

public boolean lock(String key) {

if (StringUtils.isBlank(key)) {

return false;

}

String hashCode = "" + key.hashCode();

boolean keyExist = hasKey(hashCode);

if (keyExist) {

// key存在有并发操作，锁key失败，稍后重试

return false;

} else {

// key不存在，上锁20ms

set(hashCode, "locked", 20);

return true;

}

}

//

// public static void jedis\_setex(String keyid, int i, Object list, Jedis jedis) {

// //JSON.toJSONString(list,true)

// log.info("rds setex ,keyid:"+keyid+ ",v:"+list);

// log.info(" rd ret:"+jedis.setex(keyid, 600,JSON.toJSONString(list,true)));

//

// }

public static void delX(String keyid,RedisTemplate redis) {

redis.delete(keyid);

}

@SuppressWarnings("all")

public static void setX(String key, String jsonString, int timeout,RedisTemplate redis) {

Map m=new LinkedHashMap<>();m.put("k", key);m.put("v", jsonString);m.put("timeSec", timeout);

log.info("SSSet rds ---> "+ JSON.toJSONString(m));

// RedisTemplate redis = getRedisTemplate();

// redis.opsForValue().set

redis.opsForValue().set(key, jsonString, timeout, TimeUnit.SECONDS);

}

public static String getX(String key,RedisTemplate redis) {

// RedisTemplate redis = getRedisTemplate();

return (String) redis.opsForValue().get(key );

}

public static Set<Object> zrevrange(String key, int start, int end,RedisTemplate redis) {

log.info("query rds zrevrange----key:"+key+",start:"+start+",end:"+end);

return redis .opsForZSet().reverseRange(key, start, end);

}

public static boolean zadd(String key, Object jsonString,Long getOpenTimeStamp,RedisTemplate redis) {

Map m = Maps.newLinkedHashMap();

m.put("k", key);

try {

Object obj = JSONObject.parse(jsonString.toString());

m.put("v", obj);

} catch (Exception e) {

m.put("v", jsonString);

}

m.put("score", getOpenTimeStamp);

log.info("rds zadd:" + JSON.toJSONString(m));

log.info("rds zadd readable:" + JSON.toJSONString(m, true));

// log.info("zadd ret:"+jedis.zadd(key, getOpenTimeStamp, jsonString));

try {

//log.info(redisTemplate);

//jedis.zadd(key, getOpenTimeStamp, jsonString)

log.info("zadd ret:"+ redis .opsForZSet().add(key, jsonString, getOpenTimeStamp));

return true;

} catch (Exception e) {

e.printStackTrace();

return false;

}

}

public static RedisTemplate getRedisTemplate(int DBindex) {

ResourceBundle resource = ResourceBundle.getBundle("redis");

// InputStream inStream = OpenServerController.class.getClassLoader().getResourceAsStream("redis.properties");

// Properties prop = new Properties();

// try {

// prop.load(inStream);

// } catch (IOException e) {

// throw new RuntimeException(e);

// }

String host = resource.getString("spring.redis.host");

String pwd = resource.getString("spring.redis.password");

// jedis.auth(pwd);

// jedis.select(2);

// return jedis;

RedisStandaloneConfiguration rsc = new RedisStandaloneConfiguration();

rsc.setPort(6379);

rsc.setPassword(pwd);

rsc.setHostName(host);

rsc.setDatabase(5); //jei g haosyo mayon

// JedisConnectionFactory fac = new JedisConnectionFactory(rsc);

// DefaultJedisClientConfigurationBuilder JedisClientConfiguration1=new DefaultJedisClientConfiguration();

LettuceConnectionFactory conn = new LettuceConnectionFactory(rsc);

// conn.set

// int DBindex = 7;

conn.setDatabase(DBindex); //jeig ok

// conn.setHostName(host);

// conn.setPort(6379);

// conn.setPassword(pwd);

// conn.setUsePool(true);

conn.afterPropertiesSet();

// StringRedisTemplate template

RedisTemplate<String,String> template = new RedisTemplate<>();

template.setConnectionFactory(conn);

RedisSerializer<String> redisSerializer = new StringRedisSerializer();

//key序列化方式

template.setKeySerializer(redisSerializer);

//value序列化

template.setValueSerializer(redisSerializer);

// template.setValueSerializer(StringRedisSerializer.class);

template.afterPropertiesSet();

return template;

}