Atitit mybatis 配置 redis 集成 attilax总结 艾提拉总结

[1.1. setting name="cacheEnabled" v 1](#_Toc17060)

[1.2. Mapper文件 1](#_Toc13926)

[1.3. 使用与更新 ，默认select全部使用 2](#_Toc17875)

[1.4. Redis gui 2](#_Toc28131)

[1.5. MybatisCacheWzGuava 3](#_Toc28507)

[1.6. 手动更新缓存 9](#_Toc18406)

[2. 其他增强 9](#_Toc2462)

## setting name="cacheEnabled" v

*<!DOCTYPE configuration PUBLIC "-//mybatis.org//DTD Config 3.0//EN" "http://mybatis.org/dtd/mybatis-3-config.dtd">*

<configuration>

<properties>

<property name="dialect" value="POSTGRESQL" />

<!-- <property name="dialect" value="SQLSERVER"/> -->

</properties>

<settings>

<setting name="cacheEnabled" value="true" />

## Mapper文件

<?xml version="1.0" encoding="UTF-8"?>

*<!DOCTYPE mapper PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN" "http://mybatis.org/dtd/mybatis-3-mapper.dtd">*

<mapper namespace="CliSendAdvice">

<!--

1800s 30min MybatisRedisCache

-->

<cache flushInterval="1800000" readOnly="true" size="102004" type="com.cnhis.cloudhealth.clinical.util.cache.MybatisCacheWzGuava"></cache>

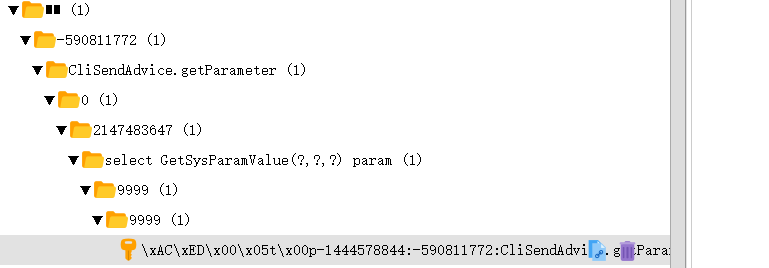
## 使用与更新 ，默认select全部使用

Select应该搞个白名单。。Update不需要，使用定期模式，因为他是根据mapper已有up操作 就清空缓存了意义不大。。

## Redis gui



\xAC\xED\x00\x05t\x00p-1444578844:-590811772:CliSendAdvice.getParameter:0:2147483647:select GetSysParamValue(?,?,?) param:9999:9999:12



这个hto d package mode is : maohao ..busi juhao ..

## MybatisCacheWzGuava

package com.cnhis.cloudhealth.clinical.util.cache;

import java.util.List;

import java.util.Map;

import java.util.concurrent.Callable;

import java.util.concurrent.FutureTask;

import java.util.concurrent.locks.ReadWriteLock;

import java.util.concurrent.locks.ReentrantReadWriteLock;

import org.apache.ibatis.cache.Cache;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

import com.cnhis.cloudhealth.clinical.util.AsynUtil\_prjcli;

import com.google.common.collect.Lists;

import redis.clients.jedis.Jedis;

import redis.clients.jedis.JedisPool;

import redis.clients.jedis.JedisPoolConfig;

// com.cnhis.cloudhealth.clinical.util.cache.MybatisRedisCache

public class MybatisRedisCache extends MybatisCacheAbs implements Cache {

public static ThreadLocal<Jedis> ThreadLocal\_jedis = new ThreadLocal<>();

public static void main(String[] args) {

}

public static String host = "192.168.1.18";

private static Logger logger = LoggerFactory.getLogger(MybatisRedisCache.class);

public static Jedis redisClient; // = createReids();

private final ReadWriteLock readWriteLock = new ReentrantReadWriteLock();

private String id;

public MybatisRedisCache(final String id) {

if (id == null) {

throw new IllegalArgumentException("Cache instances require an ID");

}

logger.debug(">>>>>>>>>>>>>>>>>>>>>>>>MybatisRedisCache:id=" + id);

this.id = id;

}

public MybatisRedisCache() {

// TODO Auto-generated constructor stub

}

@Override

public String getId() {

return this.id;

}

@Override

public int getSize() {

Jedis Jedis1 = createReids();

return Integer.valueOf(Jedis1.dbSize().toString());

}

public static ThreadLocal<Object> curkey = new ThreadLocal<>();

/\*\*

\* value arraylist

\*/

@Override

public void putObject(Object key, Object value) {

// Object curkey= MybatisRedisCache.curkey.get();

String string\_curkey = key.toString().toLowerCase();

if (string\_curkey.contains("tmp") || string\_curkey.contains("temp") || string\_curkey.contains("count")

|| string\_curkey.contains("insert") || string\_curkey.contains("update")

|| string\_curkey.contains("delete") || string\_curkey.contains("drop")

|| string\_curkey.contains("exist"))

return;

try {

logger.debug(">>>>>>>>>>>>>>>>>>>>>>>>putObject key:" + key + " ,val:" + value);

if (key == null)

return;

Jedis Jedis1 = getReids();

byte[] serialize\_key = SerializeUtil\_prjcli\_pkgcache.serialize(key.toString());

Jedis1.set(serialize\_key, SerializeUtil\_prjcli\_pkgcache.serialize(value));

try {

Jedis1.expire(serialize\_key, 300); // def 5min

} catch (Exception e) {

logger.error(" mybatisredis cache.Jedis1.expire(serialize\_key, 300) ", e);

}

} catch (Exception e) {

logger.error(" mybatisredis cache.putObject ", e);

e.printStackTrace();

}

}

@Override

public Object getObject(Object key) {

String whiteList="";

String blackList="tmp,temp,count,insert,update,delete,drop,exist";

String string\_curkey = key.toString().toLowerCase();

if (string\_curkey.contains("tmp") || string\_curkey.contains("temp") || string\_curkey.contains("count")

|| string\_curkey.contains("insert") || string\_curkey.contains("update")

|| string\_curkey.contains("delete") || string\_curkey.contains("drop")

|| string\_curkey.contains("exist"))

return null;

curkey.set(key);

logger.debug(">>>>>>>>>>>>>>>>>>>>>>>>getObject key:" + key);

Jedis Jedis1 = getReids();

try {

byte[] key2 = SerializeUtil\_prjcli\_pkgcache.serialize(key.toString());

System.out.println("--");

byte[] bytes = Jedis1.get(key2);

if (bytes == null)

return null;

Object value = SerializeUtil\_prjcli\_pkgcache.unserialize(bytes);

logger.debug(">>>>>>>>>>>>>>>>>>>>>>>>getObject key:" + key + " ,val:" + value);

return value;

} catch (Exception e) {

logger.error(" mybatisredis cache.getObject ", e);

e.printStackTrace();

}

return null;

}

@Override

public Object removeObject(Object key) {

// Jedis Jedis1= createReids();

// return

// Jedis1.expire(SerializeUtil\_prjcli\_pkgcache.serialize(key.toString()),

// 0);

return 0;

}

@Override

public void clear() {

try {

// Redis Flushdb 命令用于清空当前数据库中的所有 key。

// redisClient.flushDB();

System.out.println("");

} catch (Exception e) {

logger.error(" mybatisredis cache.clear ", e);

e.printStackTrace();

}

}

@Override

public ReadWriteLock getReadWriteLock() {

return readWriteLock;

}

protected Jedis createReids() {

/\*\*

\* Jedis jedis = new Jedis("192.168.1.18"); jedis.auth("cnhis");//

\* password return jedis;

\*/

// JedisPool pool = new JedisPool(host, 6379);

/\*

\*

\* \*/

// JedisPoolConfig config = new JedisPoolConfig();

// config.setMaxActive(500);

// config.setMaxIdle(1000);

// config.setMaxWait(1000);

// String AUTH="cnhis";

// config.setTestOnBorrow(true);config.setTestOnReturn(true);

// JedisPool JedisPool1 = new JedisPool(config, host, 6379, 5000, AUTH);

// return JedisPool1.getResource();

Jedis jedis = new Jedis("192.168.1.18");

jedis.auth("cnhis");// password

jedis.select(2);

redisClient = jedis;

return jedis;

}

public void putObjectByThrdlocalCurKey\_AsynProxy(final String key\_valideTag, final Object param\_shouldbelist) {

final Object curkey1 = MybatisRedisCache.curkey.get();

AsynUtil\_prjcli.execMeth\_Async(new FutureTask<>(new Callable() {

@Override

public Object call() throws Exception {

putObjectByThrdlocalCurKey(key\_valideTag, curkey1, param\_shouldbelist);

return null;

}

}), "threadName");

}

/\*\*

\*

\* @param param\_shouldbelist

\* @param m

\* @param parameter

\*/

synchronized public void putObjectByThrdlocalCurKey(String key\_valideTag, Object curkey1,

Object param\_shouldbelist) {

//// Object curkey = MybatisRedisCache.curkey.get();

if (curkey1 == null)

return;

if (curkey1.toString().contains("getBCK01A"))

System.out.println("dbg");

if (key\_valideTag.length() > 0)

if (!curkey1.toString().toLowerCase().contains(key\_valideTag.toLowerCase().trim()))

return;

String string\_curkey = curkey1.toString().toLowerCase();

if (string\_curkey.contains("tmp") || string\_curkey.contains("temp") || string\_curkey.contains("count")

|| string\_curkey.contains("insert") || string\_curkey.contains("update")

|| string\_curkey.contains("delete") || string\_curkey.contains("drop")

|| string\_curkey.contains("exist"))

return;

if (param\_shouldbelist == null)

return;

List param\_list = Lists.newArrayList();

if (param\_shouldbelist instanceof List) // map

{

} else

param\_list.add(param\_shouldbelist);

byte[] key2 = SerializeUtil\_prjcli\_pkgcache.serialize(string\_curkey);

Jedis Jedis1 = getReids();

if (!Jedis1.exists(key2))

putObject(curkey1, param\_list);

MybatisRedisCache.curkey.set(null);

}

private Jedis getReids() {

if (ThreadLocal\_jedis.get() == null) {

Jedis redisClient = createReids();

ThreadLocal\_jedis.set(redisClient);

}

return ThreadLocal\_jedis.get();

}

}

## 手动更新缓存

String **parameter** = cliSendAdviceDao.getParameter(map);

//ati set cache

MybatisCacheAbs1.putObjectByThrdlocalCurKey\_AsynProxy("getParameter",parameter);

// new MybatisRedisCache().putObjectByThrdlocalCurKey("getParameter",parameter);

//end set cache finish

# 其他增强