package com.arp.service.rpcloud.scheduler.service;

import com.alibaba.fastjson.JSONObject;

import com.arp.common.rpcloud.commonapi.constants.Constants;

import com.arp.common.rpcloud.commonapi.constants.exception\_code.ErrorCode;

import com.arp.common.rpcloud.commonapi.model.ServerAssets;

import com.arp.common.rpcloud.commonapi.util.RedisUtils;

import com.arp.common.rpcloud.remote.bean.ResponseBean;

import com.arp.common.rpcloud.remote.bean.scheduler.ResourceSchedulerBean;

import com.arp.common.rpcloud.remote.service.container.ContainerBusinessRemoteService;

import com.arp.common.rpcloud.remote.service.monitor.ContainerDataRemoteService;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

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

import org.springframework.stereotype.Service;

import javax.annotation.Resource;

import java.util.\*;

//import com.arp.common.rpcloud.remote.service.monitor.ContainerDataRemoteService;

/\*\*

\*

\* =======================================================

\*

\* @Version ：0.0.1

\* @Description ：资源调度service

\* ========================================================

\*/

@Service

public class SchedulerService {

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

@Autowired

private RedisUtils redisUtils;

public static String sortKeyList = "sortKeyList";

// @Autowired

// private ContainerDataRemoteService service;

private ContainerBusinessRemoteService containerBusinessRemoteService;

@Autowired

private ContainerDataRemoteService service;

@Resource(name="keyTimeOut")

private String keyTimeOut;

//@Value("${config.weightA}")

@Resource(name="weightA")

private String weightA;

@Resource(name="weightB")

private String weightB;

@Resource(name="weightC")

private String weightC;

@Resource(name="memoryTimes")

private String memoryTimes;

private static final String CPU\_ALERM\_ASSETS\_BALANCE = "cpu\_alerm\_assets\_balance";

private static final String MEMORY\_ALERM\_ASSETS\_BALANCE = "memory\_alerm\_assets\_balance";

private static final String STORAGE\_ALERM\_ASSETS\_BALANCE = "storage\_alerm\_assets\_balance";

private static final String STORAGE\_IO\_READ\_ALERM\_ASSETS\_BALANCE = "storage\_io\_read\_alerm\_assets\_balance";

private static final String STORAGE\_IO\_WRITE\_ALERM\_ASSETS\_BALANCE = "storage\_io\_write\_alerm\_assets\_balance";

private static final String NETWORK\_IO\_SEND\_ALERM\_ASSETS\_BALANCE = "network\_io\_send\_alerm\_assets\_balance";

private static final String NETWORK\_IO\_RECV\_ALERM\_ASSETS\_BALANCE = "network\_io\_recv\_alerm\_assets\_balance";

// @Autowired

// private ServerAssetsMapper serverAssetsMapper;

// @Autowired

// private SocketUtil socketUtil;

/\*\*

\* 资源调度方法

\* @Description

\* han.zhifeng

\* @param centerId 数据中心 id

\* @param cpu

\* @param memory

\* @param storageSize

\* @param imageType

\* @param isCluster

\* @param clusterMark

\* @return

\*/

public ResponseBean<ResourceSchedulerBean> resourceScheduler(String centerId, Integer cpu, Integer memory,

Integer storageSize, String imageType, String isCluster, String clusterMark){

ResponseBean<ResourceSchedulerBean> retBean = new ResponseBean<>();

String serviceId = this.getServiceId(centerId,cpu, memory);

logger.info("资源调度服务SchedulerService.resourceScheduler, 调用底层返回结果：【{}】", serviceId);

if (serviceId == null) {

logger.error("没有合适的物理机创建主机，资源紧张");

retBean.setCodeAndMsg(ErrorCode.CODE\_020101.getCode(),ErrorCode.CODE\_020101.getMsg());

return retBean;

}

ResourceSchedulerBean bean = new ResourceSchedulerBean();

bean.setServiceId(serviceId);

retBean.setData(bean);

retBean.setCodeAndMsg(ErrorCode.SUCCESS\_code.getCode(),ErrorCode.SUCCESS\_code.getMsg());

//保存上次分配的物理机,下次分配的时候,不再往这个物理机上分配

redisUtils.set("lastServerId",serviceId);

return retBean;

}

public String getServiceId(String centerId,int cpu,int memory) {

Set<String> keys = redisUtils.getKeys(Constants.schedulerData + "\*");

Map<String, Double> weightMap = new HashMap<>();

Map<String, Integer> dataMap = new HashMap<>();

for (String key : keys) {

String valueStr = redisUtils.get(key);

String serviceId = key.split("\_")[1];

int createNum = service.getTotalNumUsed(serviceId).getData();

JSONObject parse = (JSONObject) JSONObject.parse(valueStr);

Integer totalMemory = parse.getInteger("memory\_total\_num");

Integer totalCpu = parse.getInteger("cpu\_total\_num");

ResponseBean<Integer> totalCpuUsed = service.getTotalCpuUsed(serviceId);

ResponseBean<Integer> totalMemoryUsed = service.getTotalMemoryUsed(serviceId);

int canUsedMemory = totalMemory - (int)(totalMemoryUsed.getData() \* Double.parseDouble(memoryTimes)); //剩余内存

int canUsedCpu = totalCpu - totalCpuUsed.getData(); //剩余cpu

logger.info("canUsedMemory = {} ,canUsedCpu={}",canUsedMemory,canUsedCpu);

String load = parse.getString("load");

load = load.substring(1, load.length() - 2);

load = load.split(",")[0]; //拿到5分钟的负载

if (canUsedMemory > memory) {//内存够用才可以进入排序

//权重计算: memory \* A + (cpu-load)\*B - num\*C

double weight = canUsedMemory \* Double.parseDouble(weightA)

+ (canUsedCpu-Double.parseDouble(load)) \* Double.parseDouble(weightB)

- createNum \* Double.parseDouble(weightC);

weightMap.put(serviceId,weight);

dataMap.put("cpu\_" + serviceId, canUsedCpu);

dataMap.put("memory\_" + serviceId, canUsedMemory);

}

}

//对查到的信息,按照value的权重排序

List<Map.Entry<String, Double>> list = getEntriesList(weightMap);

//排完序,即使取资源最丰富的物理机,也不一定满足策略,所以验证每一项策略,去掉不符合的物理机,返回满足的物理机

List<Map.Entry<String, Double>> eligibleList = getEligibleList(centerId,cpu, memory, dataMap, list);

//可以在这里做一些策略,取第一个就是资源最丰富的物理机.也可以在队列里随机选择一个

//用random做一个伪随机,随机取一个符合条件的物理机

if(eligibleList==null) {

logger.error("没有可用资源");

return null;

}

int size = eligibleList.size();

if (size == 0) {

return null;

}else {

Random random = new Random();

int i = random.nextInt(size);

return eligibleList.get(i).getKey();

}

}

/\*\*

\* 根据策略验证每一项,去掉不符合的物理机

\* @param cpu cpu

\* @param memory memory

\* @param dataMap 数据

\* @param list 排序之后的物理机列表

\* @return

\*/

private List<Map.Entry<String, Double>> getEligibleList(String centerId, int cpu, int memory, Map<String, Integer> dataMap, List<Map.Entry<String, Double>> list) {

List<Map.Entry<String, Double>> eligibleList = new ArrayList<>();

List<ServerAssets> serverAssets = service.findAllServerAssets(centerId).getData();

for (Map.Entry<String, Double> entry : list) {

//Map.Entry<String, Integer> stringIntegerEntry = list.get(0); //资源最丰富的物理机

String mostRichKey = entry.getKey();// key 是serviceId,物理机的id

Integer mostRichCpu = dataMap.get("cpu\_" + mostRichKey);

Integer mostRichMemory = dataMap.get("memory\_" + mostRichKey);

ServerAssets theOneAssets = null;

for (ServerAssets assets : serverAssets) {

if (mostRichKey.equals(assets.getServerId())) {

theOneAssets = assets;

break;

}

}

if (theOneAssets == null) {

logger.info("数据库中未找到对应的物理机");

return null;

}

if (memory < mostRichMemory) {

//return mostRichKey;

String cpuAlermAssetsBalance = redisUtils.get("STATUS\_" + CPU\_ALERM\_ASSETS\_BALANCE + mostRichKey);

String memoryAlermAssetsBalance = redisUtils.get("STATUS\_" + MEMORY\_ALERM\_ASSETS\_BALANCE + mostRichKey);

String storageAlermAssestBalance = redisUtils.get("STATUS\_" + STORAGE\_ALERM\_ASSETS\_BALANCE + mostRichKey);

String siraab = redisUtils.get("STATUS\_" + STORAGE\_IO\_READ\_ALERM\_ASSETS\_BALANCE + mostRichKey);

String siwaab = redisUtils.get("STATUS\_" + STORAGE\_IO\_WRITE\_ALERM\_ASSETS\_BALANCE + mostRichKey);

String nisaab = redisUtils.get("STATUS\_" + NETWORK\_IO\_SEND\_ALERM\_ASSETS\_BALANCE + mostRichKey);

String niraab = redisUtils.get("STATUS\_" + NETWORK\_IO\_RECV\_ALERM\_ASSETS\_BALANCE + mostRichKey);

String lastServerId = redisUtils.get("lastServerId");

logger.info("调度条件cpuAlermAssetsBalance={},memoryAlermAssetsBalance={},storageAlermAssestBalance={}," +

"siraab={},siwaab={},nisaab={},niraab={},theOneAssets.getBalanceStatus()={},lastServerId={}",

cpuAlermAssetsBalance,

memoryAlermAssetsBalance,storageAlermAssestBalance,siraab,siwaab,nisaab,niraab,

theOneAssets.getBalanceStatus(),lastServerId);

if(!( "1".equals(cpuAlermAssetsBalance)

|| "1".equals(memoryAlermAssetsBalance)

|| "1".equals(storageAlermAssestBalance)

|| "1".equals(siraab)

|| "1".equals(siwaab)

|| "1".equals(nisaab)

|| "1".equals(niraab)

|| (theOneAssets.getBalanceStatus() != null && theOneAssets.getBalanceStatus() == 1) //是否关闭,如果关闭,不允许往这个物理机上打

|| (lastServerId!=null && lastServerId.equals(theOneAssets.getServerId())) //是否是上个物理机.如果是上个物理机,则排除

)){//if条件中有一个满足则排除,添加非!,则是满足的条件的机器

eligibleList.add(entry);

}

}else {

logger.error("可用内存不足,可用memory={},申请memory={}",mostRichMemory,memory);

}

}

logger.info("可用列表={}",eligibleList);

return eligibleList;

}

/\*\*

\* 对Map按照value进行排序

\* @param weightMap

\* @return

\*/

private List<Map.Entry<String, Double>> getEntriesList(Map<String, Double> weightMap) {

// 降序比较器(从大到小)

Comparator<Map.Entry<String, Double>> valueComparator = new Comparator<Map.Entry<String,Double>>() {

@Override

public int compare(Map.Entry<String, Double> o1,

Map.Entry<String, Double> o2) {

if(o2.getValue() > o1.getValue()) return 1;

else if(o2.getValue() < o1.getValue()) return -1;

else return 0;

}

};

// map转换成list进行排序

List<Map.Entry<String, Double>> list = new ArrayList<>(weightMap.entrySet());

// 排序

list.sort(valueComparator);

return list;

}

// /\*\*

// \* 从redis读取数据，按照指定格式再存入redis

// \*/

// @Scheduled(fixedRate = 5000)

// public void getDate() {

//

// Set<String> keys = redisUtils.getKeys(Constants.schedulerData + "\*");

// if (keys != null && keys.size() > 0) {

// logger.info("读取到心跳信息"+keys);

// String sortData = "sortData\_";

// String redisKey = sortData + System.currentTimeMillis(); //

// String orderKey = redisKey + "\_order";

//

// for (String key : keys) {

// String valueStr = redisUtils.get(key);

// //String valueStr = entry.getValue().toString();

// String serviceId = key.split("\_")[1];

// JSONObject parse = (JSONObject) JSONObject.parse(valueStr);

// Integer memoryValue = parse.getInteger("residue\_memory\_num");

// Integer cpuValue = parse.getInteger("residue\_cpu\_num");

//

// redisUtils.zadd(redisKey, memoryValue, "memory\_" + serviceId);

// redisUtils.zadd(redisKey, cpuValue, "cpu\_" + serviceId);

// redisUtils.zadd(orderKey, memoryValue + cpuValue, serviceId);

// //设置过期时间

// redisUtils.expire(redisKey, keyTimeOut);

// redisUtils.expire(orderKey, keyTimeOut);

// }

// redisUtils.lpush(sortKeyList, redisKey);

// }else {

// logger.error("没有心跳信息");

// }

//

// }

}

package com.arp.service.rpcloud.scheduler.service.redisPubSubListener;

import com.arp.common.rpcloud.commonapi.util.RedisUtils;

import com.arp.service.rpcloud.scheduler.service.SchedulerService;

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

import redis.clients.jedis.JedisPubSub;

public class DelKeyListener extends JedisPubSub {

@Autowired

RedisUtils redisUtils;

@Override

public void onSubscribe(String channel, int subscribedChannels) {

System.out.println("onSubscribe channel:" + channel + "is been subscribed:" + subscribedChannels);

}

@Override

public void onMessage(String channel, String message) {

System.out.println("onMessage channel:" + channel + "receives message :" + message);

//this.unsubscribe();

redisUtils.lrem(SchedulerService.sortKeyList, 0,message);

}

@Override

public void onPMessage(String pattern, String channel, String message) {

System.out.println("onPMessage pattern:"+pattern + " channel:"+channel+ " message:"+message);

}

}

package com.arp.service.rpcloud.scheduler.service.redisPubSubListener;

import com.arp.common.rpcloud.commonapi.util.RedisUtils;

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

import redis.clients.jedis.Jedis;

public class ListenerThread extends Thread {

@Autowired

RedisUtils redisUtils;

@Override

public void run() {

//super.run();

Jedis jedis = new Jedis("localhost",6379);

jedis.auth("B9O2yS7ZI876yxV1DeUlPpximHOs7P8v");

DelKeyListener listener = new DelKeyListener();

jedis.subscribe(listener,"\_\_keyevent@0\_\_:expired");

//jedis.psubscribe(listener,"\_\_keyevent@0\_\_:expired");

//jedis.subscribe(listener);

}

}

package com.arp.service.rpcloud.scheduler.resource;

import com.arp.common.rpcloud.remote.bean.ResponseBean;

import com.arp.common.rpcloud.remote.bean.scheduler.ResourceSchedulerBean;

import com.arp.service.rpcloud.scheduler.service.SchedulerService;

import com.arp.util.ResponseUtil;

import com.wordnik.swagger.annotations.Api;

import com.wordnik.swagger.annotations.ApiOperation;

import com.wordnik.swagger.annotations.ApiParam;

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

import org.springframework.stereotype.Component;

import javax.ws.rs.POST;

import javax.ws.rs.Path;

import javax.ws.rs.Produces;

import javax.ws.rs.QueryParam;

import javax.ws.rs.core.MediaType;

import javax.ws.rs.core.Response;

@Path("/scheduler")

@Api(value = "/scheduler", description = "资源调度底层服务", produces = MediaType.APPLICATION\_JSON)

@Produces(MediaType.APPLICATION\_JSON)

@Component

public class SchedulerResource {

@Autowired

private SchedulerService schedulerService;

@POST

@Path("/resourceScheduler")

@ApiOperation(value = "资源调度",httpMethod = "POST", notes = "用于资源调度接口")

public Response resourceScheduler(

@ApiParam(required = true, name = "centerId", value = "centerId 数据中心 id") @QueryParam("centerId") String centerId,

@ApiParam(required = true, name = "cpu", value = "cpu个数") @QueryParam("cpu") Integer cpu,

@ApiParam(required = true, name = "memory", value = "内存数量(单位为G)") @QueryParam("memory") Integer memory,

@ApiParam(required = true, name = "storageSize", value = "存储的大小（单位为MB）") @QueryParam("storageSize") Integer storageSize,

@ApiParam(required = true, name = "imageType", value = "容器镜像的类型（开发镜像develop,服务镜像service）") @QueryParam("imageType") String imageType,

@ApiParam(required = true, name = "isCluster", value = "集群标识（就是集群role 0单机 1主 2从）") @QueryParam("isCluster") String isCluster,

@ApiParam(required = true, name = "clusterMark", value = "集群的统一编码，保证一个集群的多台主机拥有相同的此参数，单机可以不传") @QueryParam("clusterMark") String clusterMark

){

ResponseBean<ResourceSchedulerBean> result = this.schedulerService.resourceScheduler(centerId, cpu, memory, storageSize, imageType, isCluster, clusterMark);

return ResponseUtil.getOkResponse(result);

}

@POST

@Path("/resourceSchedulerNullParam")

@ApiOperation(value = "资源调度",httpMethod = "POST", notes = "用于资源调度接口")

public Response resourceSchedulerNullParam(

@ApiParam(required = true, name = "centerId", value = "centerId 数据中心 id") @QueryParam("centerId") String centerId

){

ResponseBean<ResourceSchedulerBean> result = this.schedulerService.resourceScheduler(centerId,0, 0, null, null, null, null);

return ResponseUtil.getOkResponse(result);

}

}

package com.arp.service.rpcloud.scheduler.config;

import java.util.HashMap;

import java.util.Map;

import javax.servlet.Filter;

import org.springframework.boot.bind.RelaxedPropertyResolver;

import org.springframework.context.EnvironmentAware;

import org.springframework.context.annotation.Bean;

import org.springframework.context.annotation.Configuration;

import org.springframework.core.env.Environment;

import org.springframework.web.filter.CharacterEncodingFilter;

/\*\*

\*

\* 通用配置类

\*

\*/

@Configuration

public class WebConfig implements EnvironmentAware {

// 解析application.yml

private RelaxedPropertyResolver propResolver;

/\*\*

\* 解决中文内容编码问题，统一用UTF-8编码

\*

\* @return

\*/

@Bean

public Filter characterEncodingFilter() {

CharacterEncodingFilter characterEncodingFilter = new CharacterEncodingFilter();

characterEncodingFilter.setEncoding("UTF-8");

characterEncodingFilter.setForceEncoding(true);

return characterEncodingFilter;

}

@Override

public void setEnvironment(Environment environment) {

// TODO Auto-generated method stub

propResolver = new RelaxedPropertyResolver(environment);

}

public Map<String,String> staticResource(){

Map<String,String> commonStaticResource = null;

try {

commonStaticResource = new HashMap<String,String>();

commonStaticResource.put("hostUrl", propResolver.getProperty("business.hostUrl"));//医疗特殊分类“家庭医生” 前台判断业态使用

} catch (Exception e) {

e.printStackTrace();

}

return commonStaticResource;

}

}

package com.arp.service.rpcloud.scheduler.config;

import com.arp.common.rpcloud.commonapi.util.CloudProperty;

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

import org.springframework.boot.bind.RelaxedPropertyResolver;

import org.springframework.context.EnvironmentAware;

import org.springframework.context.annotation.Bean;

import org.springframework.context.annotation.Configuration;

import org.springframework.core.env.Environment;

@Configuration

public class SchedulerConfig implements EnvironmentAware {

private RelaxedPropertyResolver propResolver;

@Autowired

private CloudProperty property;

private static final String scheduler = "scheduler";

@Override

public void setEnvironment(Environment environment) {

propResolver = new RelaxedPropertyResolver(environment, "config.");

}

@Bean(name = "keyTimeOut")

public String getKeyTimeOut(){

//return Long.parseLong(propResolver.getProperty("keyTimeOut"));

return property.getProperty(scheduler,"keyTimeOut");

}

@Bean(name = "weightA")

public String getWeightA(){

//return Long.parseLong(propResolver.getProperty("keyTimeOut"));

return property.getProperty(scheduler,"weightA");

}

@Bean(name = "weightB")

public String getWeightB(){

//return Long.parseLong(propResolver.getProperty("keyTimeOut"));

return property.getProperty(scheduler,"weightB");

}

@Bean(name = "weightC")

public String getWeightC(){

//return Long.parseLong(propResolver.getProperty("keyTimeOut"));

return property.getProperty(scheduler,"weightC");

}

@Bean(name = "memoryTimes")

public String getMemoryTimes(){

//return Long.parseLong(propResolver.getProperty("keyTimeOut"));

return property.getProperty(scheduler,"memoryTimes");

}

}

package test;

import java.util.\*;

public class Test {

/\*\*

\* 对Map按照value进行排序

\* @param weightMap

\* @return

\*/

private static List<Map.Entry<String, Double>> getEntriesList(Map<String, Double> weightMap) {

// 降序比较器

Comparator<Map.Entry<String, Double>> valueComparator = new Comparator<Map.Entry<String,Double>>() {

@Override

public int compare(Map.Entry<String, Double> o1,

Map.Entry<String, Double> o2) {

// TODO Auto-generated method stub

if(o2.getValue() > o1.getValue()) return 1;

else if(o2.getValue() < o1.getValue()) return -1;

else return 0;

}

};

// map转换成list进行排序

List<Map.Entry<String, Double>> list = new ArrayList<>(weightMap.entrySet());

// 排序

list.sort(valueComparator);

return list;

}

public static void main(String[] args) {

Map<String, Double> map = new HashMap<>();

map.put("1",1.1);

map.put("2",2.0);

map.put("3",3.7);

map.put("4",4.6);

List<Map.Entry<String, Double>> entriesList = getEntriesList(map);

System.out.println(entriesList);

}

}

#!/usr/bin/env bash

# 合并dev分支到release分支

git checkout dev

git pull

git checkout offline

git merge dev -m "merge from dev"

\cp -a src/main/resources/application.yml.bak src/main/resources/application.yml

git add \*

git commit -a -m "push"

git push origin offline

buildscript {

ext {

springBootVersion = "1.5.4.RELEASE"

springCloudVersion = "Dalston.SR2"

}

repositories {

maven { url "http://10.10.10.60:8081/nexus/content/groups/public" }

maven { url "http://repo.spring.io/libs-release" }

maven { url "http://repo.spring.io/libs-milestone" }

maven { url "http://repo.spring.io/snapshot" }

jcenter()

mavenCentral()

}

dependencies {

classpath "io.spring.gradle:dependency-management-plugin:1.0.3.RELEASE"

classpath "org.springframework.boot:spring-boot-gradle-plugin:${springBootVersion}"

classpath 'org.sonarsource.scanner.gradle:sonarqube-gradle-plugin:2.0'

}

configurations {

compile.exclude module: "spring-boot-starter-tomcat"

compile.exclude module: "arp-commons-ms"

compile.exclude module: "slf4j-log4j12"

// compile.exclude module: "spring-boot-starter-jersey"

}

}

def env = System.getProperty("profile") ?: "dev"

apply plugin: "java"

apply plugin: "eclipse"

apply plugin: "idea"

apply plugin: "maven-publish"

apply plugin: "org.springframework.boot"

apply plugin: "io.spring.dependency-management"

apply plugin: 'org.sonarqube'

jar {

baseName = "fs-rpcloud-scheduler"

version = "0.0.1-SNAPSHOT"

}

// Configuration中通过package扫描注册Provider，需要在此添加对应jar包

springBoot {

requiresUnpack = ['com.wordnik:swagger-jersey2-jaxrs\_2.10', 'com.arp:arp-common']

}

sourceCompatibility = 1.8

targetCompatibility = 1.8

compileJava {

sourceCompatibility = 1.8

targetCompatibility = 1.8

options.compilerArgs = ['-parameters']

options.fork = true

options.forkOptions.executable = 'javac'

}

repositories {

mavenLocal()

maven { url "http://10.10.10.60:8081/nexus/content/groups/public" }

maven { url "http://repo.spring.io/libs-release" }

maven { url "http://repo.spring.io/libs-milestone" }

maven { url "http://repo.spring.io/snapshot" }

jcenter()

mavenCentral()

}

sonarqube {

properties {

property "sonar.sourceEncoding", "UTF-8"

}

}

dependencyManagement {

imports {

mavenBom "org.springframework.cloud:spring-cloud-starter-parent:${springCloudVersion}"

}

}

dependencies {

//springframework

compile "org.springframework.boot:spring-boot-starter-undertow"

compile "org.springframework.boot:spring-boot-starter-web"

compile "org.springframework.boot:spring-boot-starter-jersey"

compile "org.springframework.boot:spring-boot-starter-amqp"

compile "org.springframework.boot:spring-boot-starter-data-mongodb"

compile "org.springframework.boot:spring-boot-starter-actuator"

compile "org.springframework.boot:spring-boot-starter-velocity:1.3.1.RELEASE"

compile "org.springframework.boot:spring-boot-starter-mail"

compile "org.springframework.cloud:spring-cloud-starter-eureka"

compile "org.springframework.cloud:spring-cloud-starter-feign"

compile "org.springframework.cloud:spring-cloud-config-client"

//链路追踪

compile "org.springframework.cloud:spring-cloud-sleuth-zipkin-stream"

compile "org.springframework.cloud:spring-cloud-starter-sleuth"

compile "org.springframework.cloud:spring-cloud-stream-binder-rabbit"

// Database Config

compile "com.zaxxer:HikariCP"

compile "mysql:mysql-connector-java"

compile "org.mybatis.spring.boot:mybatis-spring-boot-starter:1.3.0"

// pagehelper

compile "com.github.pagehelper:pagehelper:5.0.0"

compile "com.google.guava:guava:19.0"

//common

compile "com.arp:arp-common:0.0.5-SNAPSHOT"

compile "com.arp:arp-common-redis:0.0.4-SNAPSHOT"

compile "com.arp:common-sms-api:0.0.6"

compile "org.springframework:spring-context-support:4.2.4.RELEASE"

testCompile "org.springframework.boot:spring-boot-starter-test"

compile "com.alibaba:fastjson:1.2.7"

compile 'net.java.dev.jna:jna:4.1.0'

compile ("com.arp:rpcloud-common-commonapi:0.0.2\_$env-SNAPSHOT"){

changing = true

}

compile ("com.arp:rpcloud-common-remoteservice:0.0.2\_$env-SNAPSHOT"){

changing = true

}

compile "org.springframework.data:spring-data-redis:1.8.4.RELEASE"

}

eclipse {

classpath {

containers.remove("org.eclipse.jdt.launching.JRE\_CONTAINER")

containers "org.eclipse.jdt.launching.JRE\_CONTAINER/org.eclipse.jdt.internal.debug.ui.launcher.StandardVMType/JavaSE-1.8"

downloadSources = true //eclipse下载source的配置

downloadJavadoc = false //eclipse下载doc的配置

}

}

tasks.withType(JavaCompile) {

options.encoding = "UTF-8"

}

task wrapper(type: Wrapper) {

gradleVersion = "2.9"

}

package com.arp.service.rpcloud.network.service;

import com.alibaba.fastjson.JSON;

import com.alibaba.fastjson.JSONObject;

import com.arp.common.rpcloud.commonapi.constants.Constants;

import com.arp.common.rpcloud.commonapi.mapper.\*;

import com.arp.common.rpcloud.commonapi.model.\*;

import com.arp.common.rpcloud.commonapi.util.HttpUtils;

import com.arp.common.rpcloud.remote.bean.ResponseBean;

import com.arp.common.rpcloud.remote.bean.network.BatchNetworkParams;

import com.arp.common.rpcloud.commonapi.constants.exception\_code.ErrorCode;

import com.arp.service.rpcloud.network.mapper.IpBusinessMapper;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

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

import org.springframework.stereotype.Service;

import javax.annotation.Resource;

import java.util.ArrayList;

import java.util.HashMap;

import java.util.List;

import java.util.Map;

@Service

public class NetworkBusinessService {

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

//@Value("${create.network.bridge.in}")

@Resource(name = "bridgeIn")

private String bridgeIn;

//@Value("${create.network.bridge.out}")

@Resource(name = "bridgeOut")

private String bridgeOut;

@Resource(name="userLocation")

private String userLocation;

@Resource(name = "webConsoleSshdaddr")

private String webConsoleSshdaddr;

@Resource(name = "webconsoleMaxNum")

private String webconsoleMaxNum;

@Autowired

private CloudNetworkDeviceMapper cloudNetworkDeviceMapper;

public static final String NETWORK\_TYPE\_IN = "0";

public static final String NETWORK\_TYPE\_OUT = "1";

public static final String GATEWAY\_TYPE = "2";

@Autowired

private CloudContainerAssetsMapper cloudContainerAssetsMapper;

@Autowired

private IpMapper ipMapper;

@Autowired

private PublicIpMapper publicIpMapper;

@Autowired

private NetworkContainerMacMapper networkContainerMacMapper;

@Autowired

private IpBusinessMapper ipBusinessMapper;

@Autowired

private UserInfoMapper userInfoMapper;

public ResponseBean<String> getShellBoxUrl(String containerId) {

ResponseBean<String> responseBean = new ResponseBean<>();

CloudNetworkDeviceExample example = new CloudNetworkDeviceExample();

example.createCriteria().andCloudAssetsIdEqualTo(containerId).andStatusNotEqualTo(Constants.DELETED);

List<CloudNetworkDevice> cloudNetworkDevices = cloudNetworkDeviceMapper.selectByExample(example);

for (CloudNetworkDevice cloudNetworkDevice : cloudNetworkDevices) {

if (cloudNetworkDevice.getType().equals("1")) {//外网

PublicIp publicIp = publicIpMapper.selectByPrimaryKey(cloudNetworkDevice.getIpId());

responseBean.setData(publicIp.getIp()+":4200");

responseBean.setCodeAndMsg(ErrorCode.SUCCESS\_code.getCode(),ErrorCode.SUCCESS\_code.getMsg());

return responseBean;

} else if (cloudNetworkDevice.getType().equals("0")) {

//查内网ip,获取内网网段

Ip ip = ipMapper.selectByPrimaryKey(cloudNetworkDevice.getIpId());

CloudNetworkDeviceExample example1 = new CloudNetworkDeviceExample();

example1.createCriteria().andIpIdEqualTo(ip.getIpParaId()).andStatusNotEqualTo(Constants.DELETED);

//用内网网段查出网关

List<CloudNetworkDevice> cloudNetworkDevices1 = cloudNetworkDeviceMapper.selectByExample(example1);

CloudNetworkDevice cloudNetworkDevice1 = cloudNetworkDevices1.get(0);

CloudNetworkDeviceExample example2 = new CloudNetworkDeviceExample();

example2.createCriteria().andCloudAssetsIdEqualTo(cloudNetworkDevice1.getCloudAssetsId())

.andTypeEqualTo(1);

//查出网关外网

List<CloudNetworkDevice> cloudNetworkDevices2 = cloudNetworkDeviceMapper.selectByExample(example2);

//查询外网信息

PublicIp publicIp = publicIpMapper.selectByPrimaryKey(cloudNetworkDevices2.get(0).getIpId());

responseBean.setData(publicIp.getIp()+"/"+remDot(ip.getIp()));

responseBean.setCodeAndMsg(ErrorCode.SUCCESS\_code.getCode(),ErrorCode.SUCCESS\_code.getMsg());

return responseBean;

}

}

responseBean.setData("null");

responseBean.setCodeAndMsg(ErrorCode.code\_030106.getCode(),ErrorCode.code\_030106.getMsg());

return responseBean;

}

public ResponseBean<String> getWebconsuleUrl(String containerId) {

ResponseBean<String> responseBean = new ResponseBean<>();

CloudNetworkDeviceExample example = new CloudNetworkDeviceExample();

example.createCriteria().andCloudAssetsIdEqualTo(containerId).andStatusNotEqualTo(Constants.DELETED);

List<CloudNetworkDevice> cloudNetworkDevices = cloudNetworkDeviceMapper.selectByExample(example);

for (CloudNetworkDevice cloudNetworkDevice : cloudNetworkDevices) {

if (cloudNetworkDevice.getType().equals("1")) {//外网

PublicIp publicIp = publicIpMapper.selectByPrimaryKey(cloudNetworkDevice.getIpId());

responseBean.setData(publicIp.getIp()+":22");

responseBean.setCodeAndMsg(ErrorCode.SUCCESS\_code.getCode(),ErrorCode.SUCCESS\_code.getMsg());

return responseBean;

} else if (cloudNetworkDevice.getType().equals("0")) {

Ip ip = ipMapper.selectByPrimaryKey(cloudNetworkDevice.getIpId());

responseBean.setData(ip.getIp()+":22");

responseBean.setCodeAndMsg(ErrorCode.SUCCESS\_code.getCode(),ErrorCode.SUCCESS\_code.getMsg());

return responseBean;

}

}

responseBean.setData("null");

responseBean.setCodeAndMsg(ErrorCode.code\_030106.getCode(),ErrorCode.code\_030106.getMsg());

return responseBean;

}

private String remDot(String ip) {

return ip.replaceAll("\\.","");

}

/\*\*\*

\* 组装参数

\* @param containerId

\* @return

\*/

public List<BatchNetworkParams> getPatchNetworkParams(String containerId) {

List<BatchNetworkParams> paramsList = new ArrayList<>();

//CloudContainerAssets cloudContainerAssets = cloudContainerAssetsMapper.selectByPrimaryKey(containerId);

CloudNetworkDeviceExample example = new CloudNetworkDeviceExample();

example.createCriteria().

andCloudAssetsIdEqualTo(containerId).

andStatusNotEqualTo(Constants.DELETED);

List<CloudNetworkDevice> cloudNetworkDevices = cloudNetworkDeviceMapper.selectByExample(example);

for (CloudNetworkDevice cloudNetworkDevice : cloudNetworkDevices) {

NetworkContainerMac networkContainerMac = networkContainerMacMapper.selectByPrimaryKey(cloudNetworkDevice.getInsideMacId());

if (networkContainerMac == null) {

return null;

}

BatchNetworkParams params = new BatchNetworkParams();

int type = cloudNetworkDevice.getType();

String deviceId = cloudNetworkDevice.getId();

params.setDeviceId(deviceId);

if (type == 1) {//baseNetwork

//PublicIp publicIp = publicIpMapper.selectByPrimaryKey(cloudNetworkDevice.getIpId());

List<Map<String, Object>> pubIpInfoNotEndTime = ipBusinessMapper.findPubIpInfoNotEndTime(cloudNetworkDevice.getIpId());

if (pubIpInfoNotEndTime != null && pubIpInfoNotEndTime.size() > 0) {

Map<String, Object> map = pubIpInfoNotEndTime.get(0);

params.setBridge(bridgeOut);

params.setGateway(map.get("gateway").toString());

params.setIp(map.get("ip").toString());

params.setIp\_public("1");

params.setMac(networkContainerMac.getMac());

params.setVlan("");//baseNetwork vlan is null

params.setMask("24");

String[] route = new String[1];

route[0] = map.get("gateway").toString()+"/24";

params.setRoute(route);

paramsList.add(params);

}

}else {

Map<String, Object> ipParam = new HashMap<>();

ipParam.put("ipId",cloudNetworkDevice.getIpId());

List<Map<String, Object>> ipInfo = ipBusinessMapper.findIpInfo(ipParam);

if (ipInfo == null || ipInfo.isEmpty()) {

logger.error("未找到ipId为{}的内网ip,请查询数据",cloudNetworkDevice.getIpId());

throw new RuntimeException("未找到ipId为"+cloudNetworkDevice.getIpId()+"的内网ip,请查询数据");

}

Map<String, Object> ip = ipInfo.get(0);

params.setBridge(bridgeIn);

params.setGateway(String.valueOf(ip.get("gateway")));

params.setIp(String.valueOf(ip.get("ip")));

params.setIp\_public("0");

params.setMac(networkContainerMac.getMac());

params.setVlan(String.valueOf(ip.get("vlan\_name")));

params.setMask("24");

String[] route = new String[1];

route[0] = String.valueOf(ip.get("gateway"))+"/24";

params.setRoute(route);

paramsList.add(params);

}

}

return paramsList;

}

public ResponseBean<String> getEnAddr(String vmAddr,String userId) {

UserInfoExample example = new UserInfoExample();

example.createCriteria().andUserIdEqualTo(userId);

List<UserInfo> userInfos = userInfoMapper.selectByExample(example);

UserInfo userInfo = userInfos.get(0);

String id = userInfo.getId();

int index = id.hashCode()/Integer.parseInt(webconsoleMaxNum)+1; //发送到webconsole上的序号

//http://10.10.25.39:8080/console/chksshdaddr?vm\_addr=10.10.25.45:22

//{"ok":true,"msg":"","data":{"en\_addr":"t-npNFuEf5boMzO8lyA=","sshd\_addr":"10.10.25.82:22"}

String s = HttpUtils.sendGet(webConsoleSshdaddr + "ws"+index + "/console/chksshdaddr", "vm\_addr="+vmAddr);

logger.info("s = {},webConsoleSshdaddr={},vm\_addr={}",s,webConsoleSshdaddr,vmAddr);

JSONObject jsonObject = JSON.parseObject(s);

String en\_addr = jsonObject.getJSONObject("data").getString("en\_addr");

//http://10.10.25.39/ws01/console/login/t-npNFuEf5boMzO8lyA=

String gotoUrl = webConsoleSshdaddr + "ws"+index +"/console/login/"+en\_addr;

ResponseBean<String> responseBean = new ResponseBean<>();

responseBean.setData(gotoUrl);

return responseBean;

}

}

package com.arp.service.rpcloud.network.mapper;

import org.apache.ibatis.annotations.Select;

import org.springframework.stereotype.Repository;

import java.util.List;

import java.util.Map;

/\*\*

\* 自定义mapper

\*/

@Repository

public interface IpBusinessMapper {

/\*\*

\* 通过id查找ip信息

\* @param param ipId

\* @return

\*/

@Select({

"SELECT ip.id,ip.ip,ip.mask,ip.gateway,vlan.vlan\_name ,ip.mask from ip \n" +

"LEFT JOIN intr\_ip\_para ipp ON ip.ip\_para\_id = ipp.id \n" +

"LEFT JOIN vlan ON vlan.id = ipp.vlan\_id \n" +

"where ip.id = #{ipId} "

})

List<Map<String, Object>> findIpInfo(Map<String, Object> param);

@Select({

"SELECT ip.\* from cloud\_container\_assets cca \n" +

"left JOIN cloud\_network\_device cnd on cnd.cloud\_assets\_id = cca.id\n " +

"left join ip on ip.id = cnd.ip\_id\n " +

"where cca.container\_id = #{containerId} and cnd.`status`=0 "

})

List<Map<String, Object>> findIpInfoByContainerId(Map<String, Object> param);

/\*\*

\* 通过id查找公网ip信息

\* @param ipId

\* @return

\*/

@Select({

"SELECT pubip.gateway, pubip.ip from public\_ip pubip left join public\_ip\_info pii on pubip.id = pii.public\_ip\_id\n" +

"where pubip.id = #{ipId} and NOW() < pii.end\_time "

})

List<Map<String, Object>> findPubIpInfoNotEndTime(String ipId);

}

package com.arp.service.rpcloud.assets.service;

import com.alibaba.fastjson.JSONObject;

import com.github.pagehelper.PageHelper;

import com.github.pagehelper.PageInfo;

import com.arp.common.rpcloud.commonapi.constants.Constants;

import com.arp.common.rpcloud.commonapi.constants.exception\_code.ErrorCode;

import com.arp.common.rpcloud.commonapi.mapper.ServerAssetsMapper;

import com.arp.common.rpcloud.commonapi.model.ServerAssets;

import com.arp.common.rpcloud.commonapi.model.ServerAssetsExample;

import com.arp.common.rpcloud.commonapi.util.RedisUtils;

import com.arp.common.rpcloud.remote.bean.ResponseBean;

import com.arp.common.rpcloud.remote.bean.container.ServerAssetsDataBean;

import com.arp.common.rpcloud.remote.bean.monitor.PhysicalMonitorData;

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

import org.springframework.stereotype.Service;

import java.util.ArrayList;

import java.util.List;

import java.util.Map;

/\*\*

\* =======================================================

\*

\* @Version :0.0.1

\* @Description : 资产管理

\* ========================================================

\*/

@Service

public class AssetManagementService {

@Autowired

private ServerAssetsMapper service;

/\*\*

\* 查询数据　（分页）

\*

\* @param assetsId

\* @param adminName

\* @param tele

\* @param page

\* @param pageNum

\* @return

\*/

public ResponseBean<PageInfo<ServerAssetsDataBean>> queryAsset(String assetsId, String adminName, String tele, Integer page, Integer pageNum) {

ResponseBean<PageInfo<ServerAssetsDataBean>> responseBean = new ResponseBean<>();

try {

PageHelper.startPage(page, pageNum);

ServerAssetsExample ame = new ServerAssetsExample();

ServerAssetsExample.Criteria criteria = ame.createCriteria();

if (null != assetsId && !assetsId.isEmpty()) {

// criteria.andServerIdLike("%" + assetsId + "%");

criteria.andIpLike("%" + assetsId + "%");

}

if (null != adminName && !adminName.isEmpty()) {

criteria.andAdminNameLike("%" + adminName + "%");

}

if (null != tele && !tele.isEmpty()) {

criteria.andPhoneLike("%" + tele + "%");

}

List<ServerAssets> list = service.selectByExample(ame);

PageInfo<ServerAssets> pageInfo = new PageInfo<>(list);

List<ServerAssetsDataBean> newList = new ArrayList<>();

for (ServerAssets serverAssets : list) {

ServerAssetsDataBean serverAssetsDataBean = new ServerAssetsDataBean();

serverAssetsDataBean.setServerAssets(serverAssets);

if (serverAssets.getServerId() != null) {

serverAssetsDataBean.setPhysicalMonitorData(getServerAssetsConfig(serverAssets.getServerId()));

}

newList.add(serverAssetsDataBean);

}

PageInfo<ServerAssetsDataBean> newPageInfo = new PageInfo();

newPageInfo.setPageNum(pageInfo.getPageNum());

newPageInfo.setPageSize(pageInfo.getPageSize());

newPageInfo.setPages(pageInfo.getPages());

newPageInfo.setList(newList);

newPageInfo.setSize(pageInfo.getSize());

newPageInfo.setTotal(pageInfo.getTotal());

responseBean.setCodeAndMsg(ErrorCode.SUCCESS.getCode(), ErrorCode.SUCCESS.getMsg());

responseBean.setData(newPageInfo);

} catch (Exception e) {

responseBean.setCodeAndMsg(ErrorCode.CODE\_100901.getCode(), ErrorCode.CODE\_100901.getMsg());

}

return responseBean;

}

/\*\*

\* 查询数据

\*

\* @param id

\* @return

\*/

public ResponseBean<ServerAssetsDataBean> queryAssetById(String id) {

ResponseBean<ServerAssetsDataBean> responseBean = new ResponseBean<>();

try {

ServerAssetsDataBean serverAssetsDataBean = new ServerAssetsDataBean();

ServerAssets am = service.selectByPrimaryKey(id);

serverAssetsDataBean.setServerAssets(am);

serverAssetsDataBean.setPhysicalMonitorData(getServerAssetsConfig(am.getServerId()));

responseBean.setCodeAndMsg(ErrorCode.SUCCESS.getCode(), ErrorCode.SUCCESS.getMsg());

responseBean.setData(serverAssetsDataBean);

} catch (Exception e) {

responseBean.setCodeAndMsg(ErrorCode.CODE\_100901.getCode(), ErrorCode.CODE\_100901.getMsg());

}

return responseBean;

}

/\*\*

\* 　添加数据

\*

\* @param serverAssets

\* @return

\*/

public ResponseBean addAsset(ServerAssets serverAssets) {

ResponseBean responseBean = new ResponseBean<>();

try {

int insNumber = service.insert(serverAssets);

responseBean.setCodeAndMsg(ErrorCode.SUCCESS.getCode(), ErrorCode.SUCCESS.getMsg());

} catch (Exception e) {

responseBean.setCodeAndMsg(ErrorCode.CODE\_100901.getCode(), ErrorCode.CODE\_100901.getMsg());

}

return responseBean;

}

/\*\*

\* 修改数据

\*

\* @param serverAssets

\* @return

\*/

public ResponseBean updateAsset(ServerAssets serverAssets) {

ResponseBean responseBean = new ResponseBean();

try {

int updateNumber = service.updateByPrimaryKeySelective(serverAssets);//updateByPrimaryKey(serverAssets);

responseBean.setCodeAndMsg(ErrorCode.SUCCESS.getCode(), ErrorCode.SUCCESS.getMsg());

} catch (Exception e) {

responseBean.setCodeAndMsg(ErrorCode.CODE\_100902.getCode(), ErrorCode.CODE\_100902.getMsg());

}

return responseBean;

}

/\*\*

\* 删除数据

\*

\* @param id

\* @return

\*/

public ResponseBean<Boolean> deleteAsset(String id) {

ResponseBean<Boolean> responseBean = new ResponseBean<>();

try {

int updateNumber = service.deleteByPrimaryKey(id);

responseBean.setCodeAndMsg(ErrorCode.SUCCESS.getCode(), ErrorCode.SUCCESS.getMsg());

responseBean.setData(true);

} catch (Exception e) {

responseBean.setCodeAndMsg(ErrorCode.CODE\_100903.getCode(), ErrorCode.CODE\_100903.getMsg());

}

return responseBean;

}

@Autowired

RedisUtils redisUtils;

/\*\*

\* 读取资产信息

\*

\* @param serverId

\* @return

\*/

private PhysicalMonitorData getServerAssetsConfig(String serverId) {

String dataStr = redisUtils.get(Constants.PHYSICAL\_MONITOR\_NODE + serverId);

if (dataStr != null) {

JSONObject json = JSONObject.parseObject(dataStr);

return JSONObject.toJavaObject(json, PhysicalMonitorData.class);

}

return null;

}

public ResponseBean<Boolean> updateServerAssetsBalanceStatus(String serverAssetsBalance, Integer balanceStatus) {

ResponseBean<Boolean> responseBean = new ResponseBean<>();

ServerAssets serverAssets = new ServerAssets();

serverAssets.setId(serverAssetsBalance);

serverAssets.setBalanceStatus(balanceStatus);

service.updateByPrimaryKeySelective(serverAssets);

responseBean.setCodeAndMsg(ErrorCode.SUCCESS.getCode(), ErrorCode.SUCCESS.getMsg());

return responseBean;

}

}

package com.arp.service.rpcloud.assets.service;

import com.alibaba.fastjson.JSONObject;

import com.github.pagehelper.PageHelper;

import com.github.pagehelper.PageInfo;

import com.arp.common.rpcloud.commonapi.constants.exception\_code.ErrorCode;

import com.arp.common.rpcloud.commonapi.model.ServerCenter;

import com.arp.common.rpcloud.commonapi.model.ServerModule;

import com.arp.common.rpcloud.commonapi.model.ServerRack;

import com.arp.common.rpcloud.commonapi.model.ServerRegion;

import com.arp.common.rpcloud.commonapi.util.MD5;

import com.arp.common.rpcloud.remote.bean.ResponseBean;

import com.arp.service.rpcloud.assets.mapper.PhysicalMachineMagMapper;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

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

import org.springframework.stereotype.Service;

import org.springframework.util.ObjectUtils;

import java.util.ArrayList;

import java.util.Date;

import java.util.List;

import java.util.Map;

/\*\*

\* Created by wugn on 2018/3/29.

\*/

@Service

public class PhysicalMachineMagService {

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

//临时记录某一区域下的数据中心的所有id

private List<String> listCenterId = null;

//临时记录某一数据中心下的模块的所有id

private List<String> listModuleId = null;

//临时记录某一模块下的机架的所有id

private List<String> listRackId = null;

@Autowired

PhysicalMachineMagMapper physicalMachineMagMapper;

/\*\*

\* 创建区域

\* @param serverRegion

\* @return

\*/

public ResponseBean insertRegion(ServerRegion serverRegion){

ResponseBean<Boolean> responseBean = new ResponseBean<>();

try {

logger.info("创建物理机区域-参数:"+serverRegion);

serverRegion.setId(MD5.uuid());

serverRegion.setCreateTime(new Date());

serverRegion.setUpdateTime(new Date());

physicalMachineMagMapper.insertRegion(serverRegion);

responseBean.setCodeAndMsg(ErrorCode.SUCCESS.getCode(), ErrorCode.SUCCESS.getMsg());

}catch (Exception e){

logger.info("添加区域失败!");

responseBean.setCodeAndMsg(ErrorCode.CODE\_120101.getCode(),ErrorCode.CODE\_120101.getMsg());

}

return responseBean;

}

/\*\*

\* 分页查询区域

\* @param object

\* @return

\*/

public ResponseBean<PageInfo<Map<String, Object>>> selectRegion(JSONObject object){

ResponseBean<PageInfo<Map<String, Object>>> responseBean = new ResponseBean();

try {

logger.info("查询物理机区域-参数:"+object.toString());

Integer pageNum = Integer.valueOf(object.get("pageNum").toString());

Integer pageSize = Integer.valueOf(object.get("pageSize").toString());

PageHelper.startPage(pageNum, pageSize);

List<Map<String, Object>> list = physicalMachineMagMapper.selectRegion();

PageInfo<Map<String, Object>> pageInfo = new PageInfo<>(list);

responseBean.setData(pageInfo);

responseBean.setCodeAndMsg(ErrorCode.SUCCESS.getCode(), ErrorCode.SUCCESS.getMsg());

}catch (Exception e){

logger.error("查询物理机区域失败!");

responseBean.setCodeAndMsg(ErrorCode.CODE\_120102.getCode(),ErrorCode.CODE\_120102.getMsg());

}

return responseBean;

}

/\*\*

\* 修改区域信息

\* @param serverRegion

\* @return

\*/

public ResponseBean updateRegion(ServerRegion serverRegion){

ResponseBean<Boolean> responseBean = new ResponseBean<>();

try {

logger.info("修改物理机区域-参数:"+serverRegion);

serverRegion.setUpdateTime(new Date());

physicalMachineMagMapper.updateRegion(serverRegion);

responseBean.setCodeAndMsg(ErrorCode.SUCCESS.getCode(), ErrorCode.SUCCESS.getMsg());

}catch (Exception e){

logger.error("修改区域失败!");

responseBean.setCodeAndMsg(ErrorCode.CODE\_120103.getCode(),ErrorCode.CODE\_120103.getMsg());

}

return responseBean;

}

/\*\*

\* 删除分区

\* @param object

\* @return

\*/

public ResponseBean deleteRegion(JSONObject object){

ResponseBean<Boolean> responseBean = new ResponseBean<>();

try {

logger.info("删除物理机区域-参数:"+object.toString());

String regionId = object.getString("id");

List list = physicalMachineMagMapper.regionIsAssets(regionId);

if(!ObjectUtils.isEmpty(list)){

responseBean.setCodeAndMsg(ErrorCode.CODE\_120105.getCode(),ErrorCode.CODE\_120105.getMsg());

}else {

List<Map<String, Object>> centerList = physicalMachineMagMapper.regionIsCenter(regionId);

physicalMachineMagMapper.deleteRegion(regionId);

listCenterId = new ArrayList<>();

if(!ObjectUtils.isEmpty(centerList)) {

for (Map<String, Object> map : centerList) {

String c\_id = map.get("id").toString();

listCenterId.add(c\_id);

}

}

this.delete\_Center();

this.delete\_Module();

this.delete\_Rack();

}

responseBean.setCodeAndMsg(ErrorCode.SUCCESS.getCode(), ErrorCode.SUCCESS.getMsg());

}catch (Exception e){

logger.error("删除区域失败!");

responseBean.setCodeAndMsg(ErrorCode.CODE\_120104.getCode(),ErrorCode.CODE\_120104.getMsg());

}

return responseBean;

}

/\*\*

\* 创建数据中心

\* @param serverCenter

\* @return

\*/

public ResponseBean insertCenter(ServerCenter serverCenter){

ResponseBean<Boolean> responseBean = new ResponseBean<>();

try {

logger.info("创建物理机数据中心-参数:"+serverCenter);

serverCenter.setId(MD5.uuid());

serverCenter.setCreateTime(new Date());

serverCenter.setUpdateTime(new Date());

physicalMachineMagMapper.insertCenter(serverCenter);

responseBean.setCodeAndMsg(ErrorCode.SUCCESS.getCode(), ErrorCode.SUCCESS.getMsg());

}catch (Exception e){

logger.error("创建数据中心失败!");

responseBean.setCodeAndMsg(ErrorCode.CODE\_120201.getCode(),ErrorCode.CODE\_120201.getMsg());

}

return responseBean;

}

/\*\*

\* 分页查询 数据中心

\* @param object

\* @return

\*/

public ResponseBean<PageInfo<Map<String, Object>>> selectCenter(JSONObject object){

ResponseBean<PageInfo<Map<String, Object>>> responseBean = new ResponseBean();

try {

logger.info("查询数据中心-参数:"+object.toString());

Integer pageNum = Integer.valueOf(object.get("pageNum").toString());

Integer pageSize = Integer.valueOf(object.get("pageSize").toString());

String regionId = object.getString("regionId");

PageHelper.startPage(pageNum, pageSize);

List<Map<String, Object>> list = physicalMachineMagMapper.selectCenter(regionId);

PageInfo<Map<String, Object>> pageInfo = new PageInfo<>(list);

responseBean.setData(pageInfo);

responseBean.setCodeAndMsg(ErrorCode.SUCCESS.getCode(), ErrorCode.SUCCESS.getMsg());

}catch (Exception e){

logger.error("查询数据中心失败!");

responseBean.setCodeAndMsg(ErrorCode.CODE\_120202.getCode(),ErrorCode.CODE\_120202.getMsg());

}

return responseBean;

}

/\*\*

\* 修改数据中心

\* @param serverCenter

\* @return

\*/

public ResponseBean updateCenter(ServerCenter serverCenter){

ResponseBean<Boolean> responseBean = new ResponseBean<>();

try {

logger.info("修改数据中心-参数:"+serverCenter);

serverCenter.setUpdateTime(new Date());

physicalMachineMagMapper.updateCenter(serverCenter);

responseBean.setCodeAndMsg(ErrorCode.SUCCESS.getCode(), ErrorCode.SUCCESS.getMsg());

}catch (Exception e){

logger.error("修改数据中心失败!");

responseBean.setCodeAndMsg(ErrorCode.CODE\_120203.getCode(),ErrorCode.CODE\_120203.getMsg());

}

return responseBean;

}

/\*\*

\* 删除数据中心

\* @param object

\* @return

\*/

public ResponseBean deleteCenter(JSONObject object){

ResponseBean<Boolean> responseBean = new ResponseBean<>();

try {

logger.info("删除数据中心-参数:"+object.toString());

String centerId = object.getString("centerId");

List list = physicalMachineMagMapper.centerIsAssets(centerId);

if(!ObjectUtils.isEmpty(list)){

responseBean.setCodeAndMsg(ErrorCode.CODE\_120205.getCode(),ErrorCode.CODE\_120205.getMsg());

}else {

List<Map<String, Object>> moduleList = physicalMachineMagMapper.centerIsModule(centerId);

physicalMachineMagMapper.deleteCenter(centerId);

listModuleId = new ArrayList<>();

if(!ObjectUtils.isEmpty(moduleList)){

for (Map<String, Object> map : moduleList) {

String m\_id = map.get("id").toString();

listModuleId.add(m\_id);

}

}

this.delete\_Module();

this.delete\_Rack();

}

responseBean.setCodeAndMsg(ErrorCode.SUCCESS.getCode(), ErrorCode.SUCCESS.getMsg());

}catch (Exception e){

logger.error("删除区域失败!");

responseBean.setCodeAndMsg(ErrorCode.CODE\_120204.getCode(),ErrorCode.CODE\_120203.getMsg());

}

return responseBean;

}

/\*\*

\* 创建模块

\* @param serverModule

\* @return

\*/

public ResponseBean insertModule(ServerModule serverModule){

ResponseBean<Boolean> responseBean = new ResponseBean<>();

try {

logger.info("创建模块-参数:"+serverModule);

serverModule.setId(MD5.uuid());

serverModule.setCreateTime(new Date());

serverModule.setUpdateTime(new Date());

}catch (Exception e){

logger.error("创建模块失败!");

responseBean.setCodeAndMsg(ErrorCode.CODE\_120301.getCode(),ErrorCode.CODE\_120301.getMsg());

}

return responseBean;

}

/\*\*

\* 分页查询 模块

\* @param object

\* @return

\*/

public ResponseBean<PageInfo<Map<String, Object>>> selectModule(JSONObject object){

ResponseBean<PageInfo<Map<String, Object>>> responseBean = new ResponseBean();

try {

logger.info("查询模块-参数:"+object.toString());

Integer pageNum = Integer.valueOf(object.get("pageNum").toString());

Integer pageSize = Integer.valueOf(object.get("pageSize").toString());

String centerId = object.getString("centerId");

PageHelper.startPage(pageNum, pageSize);

List<Map<String, Object>> list = physicalMachineMagMapper.selectModule(centerId);

PageInfo<Map<String, Object>> pageInfo = new PageInfo<>(list);

responseBean.setData(pageInfo);

responseBean.setCodeAndMsg(ErrorCode.SUCCESS.getCode(), ErrorCode.SUCCESS.getMsg());

}catch (Exception e){

logger.error("查询模块失败!");

responseBean.setCodeAndMsg(ErrorCode.CODE\_120302.getCode(),ErrorCode.CODE\_120302.getMsg());

}

return responseBean;

}

/\*\*

\* 修改模块

\* @param serverModule

\* @return

\*/

public ResponseBean updateModule(ServerModule serverModule){

ResponseBean<Boolean> responseBean = new ResponseBean<>();

try {

logger.info("修改模块-参数:"+serverModule);

serverModule.setUpdateTime(new Date());

physicalMachineMagMapper.updateModule(serverModule);

responseBean.setCodeAndMsg(ErrorCode.SUCCESS.getCode(), ErrorCode.SUCCESS.getMsg());

}catch (Exception e){

logger.error("修改模块失败!");

responseBean.setCodeAndMsg(ErrorCode.CODE\_120304.getCode(),ErrorCode.CODE\_120304.getMsg());

}

return responseBean;

}

/\*\*

\* 删除模块

\* @param object

\* @return

\*/

public ResponseBean deleteModule(JSONObject object){

ResponseBean<Boolean> responseBean = new ResponseBean<>();

try {

logger.info("删除模块-参数:"+object.toString());

String moduleId = object.getString("moduleId");

List list = physicalMachineMagMapper.moduleIsAssets(moduleId);

if(!ObjectUtils.isEmpty(list)){

responseBean.setCodeAndMsg(ErrorCode.CODE\_120305.getCode(),ErrorCode.CODE\_120305.getMsg());

}else {

List<Map<String, Object>> rackList = physicalMachineMagMapper.moduleIsRack(moduleId);

physicalMachineMagMapper.deleteModule(moduleId);

listRackId = new ArrayList<>();

if(!ObjectUtils.isEmpty(rackList)){

for (Map<String, Object> map : rackList) {

String m\_id = map.get("id").toString();

listRackId.add(m\_id);

}

}

this.delete\_Rack();

}

responseBean.setCodeAndMsg(ErrorCode.SUCCESS.getCode(), ErrorCode.SUCCESS.getMsg());

}catch (Exception e){

logger.error("删除模块失败!");

responseBean.setCodeAndMsg(ErrorCode.CODE\_120304.getCode(),ErrorCode.CODE\_120304.getMsg());

}

return responseBean;

}

/\*\*

\* 添加机架

\* @param serverRack

\* @return

\*/

public ResponseBean insertRack(ServerRack serverRack){

ResponseBean<Boolean> responseBean = new ResponseBean<>();

try{

logger.info("创建机架-参数:"+serverRack);

serverRack.setId(MD5.uuid());

serverRack.setCreateTime(new Date());

serverRack.setUpdateTime(new Date());

physicalMachineMagMapper.insertRack(serverRack);

}catch (Exception e){

logger.error("创建机架失败!");

responseBean.setCodeAndMsg(ErrorCode.CODE\_120401.getCode(),ErrorCode.CODE\_120401.getMsg());

}

return responseBean;

}

/\*\*

\* 分页查询 机架

\* @param object

\* @return

\*/

public ResponseBean<PageInfo<Map<String, Object>>> selectRack(JSONObject object){

ResponseBean<PageInfo<Map<String, Object>>> responseBean = new ResponseBean();

try {

logger.info("查询模块-参数:"+object.toString());

Integer pageNum = Integer.valueOf(object.get("pageNum").toString());

Integer pageSize = Integer.valueOf(object.get("pageSize").toString());

String moduleId = object.getString("moduleId");

PageHelper.startPage(pageNum, pageSize);

List<Map<String, Object>> list = physicalMachineMagMapper.selectRack(moduleId);

PageInfo<Map<String, Object>> pageInfo = new PageInfo<>(list);

responseBean.setData(pageInfo);

responseBean.setCodeAndMsg(ErrorCode.SUCCESS.getCode(), ErrorCode.SUCCESS.getMsg());

}catch (Exception e){

logger.error("查询模块失败!");

responseBean.setCodeAndMsg(ErrorCode.CODE\_120402.getCode(),ErrorCode.CODE\_120402.getMsg());

}

return responseBean;

}

/\*\*

\* 修改模块

\* @param serverRack

\* @return

\*/

public ResponseBean updateRack(ServerRack serverRack){

ResponseBean<Boolean> responseBean = new ResponseBean<>();

try {

logger.info("修改机架-参数:"+ serverRack);

serverRack.setUpdateTime(new Date());

physicalMachineMagMapper.updateRack(serverRack);

responseBean.setCodeAndMsg(ErrorCode.SUCCESS.getCode(), ErrorCode.SUCCESS.getMsg());

}catch (Exception e){

logger.error("修改机架失败!");

responseBean.setCodeAndMsg(ErrorCode.CODE\_120403.getCode(),ErrorCode.CODE\_120403.getMsg());

}

return responseBean;

}

/\*\*

\* 删除机架

\* @param object

\* @return

\*/

public ResponseBean deleteRack(JSONObject object){

ResponseBean<Boolean> responseBean = new ResponseBean<>();

try {

logger.info("删除机架-参数:"+object.toString());

String rackId = object.getString("rackId");

List list = physicalMachineMagMapper.rackIsAssets(rackId);

if(!ObjectUtils.isEmpty(list)){

responseBean.setCodeAndMsg(ErrorCode.CODE\_120405.getCode(),ErrorCode.CODE\_120405.getMsg());

}else {

physicalMachineMagMapper.deleteRack(rackId);

}

responseBean.setCodeAndMsg(ErrorCode.SUCCESS.getCode(), ErrorCode.SUCCESS.getMsg());

}catch (Exception e){

logger.error("删除机架失败!");

responseBean.setCodeAndMsg(ErrorCode.CODE\_120404.getCode(),ErrorCode.CODE\_120404.getMsg());

}

return responseBean;

}

/\*\*

\* 抽象 数据中心删除

\*/

private void delete\_Center(){

for(String id : listCenterId){

List<Map<String, Object>> moduleList = physicalMachineMagMapper.centerIsModule(id);

physicalMachineMagMapper.deleteCenter(id);

listModuleId = new ArrayList<>();

if(!ObjectUtils.isEmpty(moduleList)){

for (Map<String, Object> map : moduleList) {

String m\_id = map.get("id").toString();

listModuleId.add(m\_id);

}

}

}

}

/\*\*

\* 抽象 模块删除

\*/

private void delete\_Module(){

for(String id : listModuleId){

List<Map<String, Object>> rackList = physicalMachineMagMapper.moduleIsRack(id);

physicalMachineMagMapper.deleteModule(id);

listRackId = new ArrayList<>();

if(!ObjectUtils.isEmpty(rackList)){

for (Map<String, Object> map : rackList) {

String r\_id = map.get("id").toString();

listRackId.add(r\_id);

}

}

}

}

/\*\*

\* 抽象 机架删除

\*/

private void delete\_Rack(){

for (String id : listRackId){

physicalMachineMagMapper.deleteRack(id);

}

}

}

package com.arp.service.rpcloud.network.resource;

import java.util.HashMap;

import java.util.List;

import java.util.Map;

import javax.ws.rs.GET;

import javax.ws.rs.Path;

import javax.ws.rs.Produces;

import javax.ws.rs.QueryParam;

import javax.ws.rs.core.MediaType;

import org.slf4j.Logger;

import org.slf4j.LoggerFactory;

import org.springframework.amqp.rabbit.core.RabbitTemplate;

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

import org.springframework.stereotype.Component;

import com.arp.common.rpcloud.commonapi.bean.NoticeBean;

import com.arp.common.rpcloud.commonapi.constants.Constants;

import com.arp.common.rpcloud.commonapi.util.RedisUtils;

import com.arp.service.rpcloud.network.mapper.TableToMapper;

import com.wordnik.swagger.annotations.Api;

import com.wordnik.swagger.annotations.ApiOperation;

import com.wordnik.swagger.annotations.ApiParam;

@Path("/tableToMapper")

@Api(value = "/tableToMapper", description = "表tobean", produces = MediaType.APPLICATION\_JSON)

@Produces(MediaType.APPLICATION\_JSON)

@Component

public class TableToMapperResource {

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

@Autowired

TableToMapper mapper;

@Autowired

RedisUtils redisUtils;

@Autowired

private RabbitTemplate rabbitTemplate;

@GET

@Path("/to")

@ApiOperation(value = "创建安全组", httpMethod = "GET", notes = "创建安全组")

public void to(

) {

List<Map<String, Object>> list = this.mapper.getTables();

for (Map<String, Object> map : list) {

String key = map.get("Tables\_in\_arp\_cloud\_3").toString();

String[] arr = key.split("\_");

String two = "";

for (String s : arr) {

two = two + s.substring(0, 1).toUpperCase() + s.substring(1);

}

String to = "<table tableName=\"" + key + "\" domainObjectName=\"" + two + "\" />";

System.out.println(to);

}

}

@GET

@Path("/getProperties")

@ApiOperation(value = "getredis", httpMethod = "GET", notes = "getredis")

public Map<String, Object> getProperties(

@ApiParam(required = true, name = "properties", value = "用户id") @QueryParam("properties") String properties

) {

String value = this.redisUtils.get("CloudProperty." + properties);

Map<String, Object> map = new HashMap<>();

map.put("value", value);

logger.info("----------");

logger.info(value);

return map;

}

@GET

@Path("/getKey")

@ApiOperation(value = "getredis", httpMethod = "GET", notes = "getredis")

public Map<String, Object> getKey(

@ApiParam(required = true, name = "properties", value = "用户id") @QueryParam("properties") String properties

) {

String value = this.redisUtils.get("CloudProperty." + properties);

Map<String, Object> map = new HashMap<>();

map.put("value", value);

logger.info("----------");

logger.info(value);

return map;

}

@GET

@Path("/deleteProperties")

@ApiOperation(value = "getredis", httpMethod = "GET", notes = "getredis")

public void deleteProperties(

@ApiParam(required = true, name = "properties", value = "用户id") @QueryParam("properties") String properties

) {

this.redisUtils.del("CloudProperty." + properties);

logger.info("----------");

}

@GET

@Path("/testWebsocket")

@ApiOperation(value = "getredis", httpMethod = "GET", notes = "getredis")

public String testWebsocket(

@ApiParam(required = true, name = "userId", value = "用户id") @QueryParam("userId") String userId,

@ApiParam(required = true, name = "activeType", value = "类型") @QueryParam("activeType") Integer activeType,

@ApiParam(required = true, name = "content", value = "内容") @QueryParam("content") String content

) {

logger.info("测试websocket");

NoticeBean nb = new NoticeBean();

nb.setUserId(userId);

nb.setActiveType(activeType);

nb.setRetCode("000000");

nb.setRetMessage(content);

logger.info("发送：【{}】", nb.toJson());

rabbitTemplate.convertAndSend(Constants.NOTICE\_WEB\_QUEUE,"", nb.toJson());

return "测试websocket";

}

}

package com.arp.service.rpcloud.network.resource;

import java.util.List;

import java.util.Map;

import javax.ws.rs.POST;

import javax.ws.rs.Path;

import javax.ws.rs.Produces;

import javax.ws.rs.QueryParam;

import javax.ws.rs.core.MediaType;

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

import org.springframework.stereotype.Component;

import org.springframework.web.bind.annotation.RequestBody;

import org.springframework.web.bind.annotation.RequestParam;

import com.alibaba.fastjson.JSONObject;

import com.github.pagehelper.PageInfo;

import com.arp.common.rpcloud.commonapi.model.AppGroup;

import com.arp.common.rpcloud.commonapi.model.CloudContainerAssets;

import com.arp.common.rpcloud.commonapi.model.Cluster;

import com.arp.common.rpcloud.commonapi.model.NetworkSecurityGroup;

import com.arp.common.rpcloud.commonapi.model.NetworkSecurityGroupTag;

import com.arp.common.rpcloud.remote.bean.ResponseBean;

import com.arp.service.rpcloud.network.service.SecurityGroupBusinessService;

import com.wordnik.swagger.annotations.Api;

import com.wordnik.swagger.annotations.ApiOperation;

import com.wordnik.swagger.annotations.ApiParam;

@Path("/securityGroupBusiness")

@Api(value = "/securityGroupBusiness", description = "安全组服务", produces = MediaType.APPLICATION\_JSON)

@Produces(MediaType.APPLICATION\_JSON)

@Component

public class SecurityGroupBusinessResource {

@Autowired

private SecurityGroupBusinessService securityGroupBusinessService;

@POST

@Path("/addSecurityGroup")

@ApiOperation(value = "创建安全组", httpMethod = "POST", notes = "创建安全组")

public ResponseBean<Void> addSecurityGroup(

@ApiParam(required = true, name = "obj", value = "安全组参数") @RequestBody JSONObject obj,

@ApiParam(required = true, name = "userId", value = "用户id") @QueryParam("userId") String userId

) {

return securityGroupBusinessService.addSecurityGroup(obj, userId);

}

@POST

@Path("/addGroupRole")

@ApiOperation(value = "给安全组添加规则", httpMethod = "POST", notes = "给安全组添加规则")

public ResponseBean<Void> addGroupRole(

@ApiParam(required = true, name = "obj", value = "安全组参数") @RequestBody JSONObject obj

) {

return securityGroupBusinessService.addGroupRole(obj);

}

@POST

@Path("/deleteGroupRole")

@ApiOperation(value = "给安全组删除规则", httpMethod = "POST", notes = "给安全组删除规则")

public ResponseBean<Void> deleteGroupRole(

@ApiParam(required = true, name = "obj", value = "安全组参数") @RequestBody JSONObject obj

) {

return securityGroupBusinessService.deleteGroupRole(obj);

}

@POST

@Path("/editGroupRole")

@ApiOperation(value = "编辑安全组规则", httpMethod = "POST", notes = "编辑安全组规则")

public ResponseBean<Void> editGroupRole(

@ApiParam(required = true, name = "obj", value = "安全组参数") @RequestBody JSONObject obj

) {

return securityGroupBusinessService.editGroupRole(obj);

}

@POST

@Path("/bindSecurityGroup")

@ApiOperation(value = "绑定安全组", httpMethod = "POST", notes = "绑定安全组")

public ResponseBean<Void> bindSecurityGroup(

@ApiParam(required = true, name = "obj", value = "安全组参数") @RequestBody JSONObject obj

) {

return securityGroupBusinessService.bindSecurityGroup(obj);

}

@POST

@Path("/bindIpForGroupActive")

@ApiOperation(value = "ip绑定安全组回调", httpMethod = "POST", notes = "ip绑定安全组回调")

public void bindIpForGroupActive(

@ApiParam(required = true, name = "obj", value = "安全组参数") @RequestBody JSONObject obj

) {

securityGroupBusinessService.bindIpForGroupActive(obj);

}

@POST

@Path("/unBindIpGroup")

@ApiOperation(value = "解绑安全组", httpMethod = "POST", notes = "解绑安全组")

public ResponseBean<Void> unBindSecurityGroup(

@ApiParam(required = true, name = "obj", value = "解绑ip的安全组") @RequestBody JSONObject obj

) {

return securityGroupBusinessService.unBindSecurityGroup(obj);

}

@POST

@Path("/unBindIpGroupActive")

@ApiOperation(value = "删除完全组回调", httpMethod = "POST", notes = "删除完全组回调")

public void unBindIpGroupActive(

@ApiParam(required = true, name = "obj", value = "删除完全组回调") @RequestBody JSONObject obj

) {

securityGroupBusinessService.unBindIpGroupActive(obj);

}

@POST

@Path("/getUserPriority")

@ApiOperation(value = "获取用户可用的优先级", httpMethod = "POST", notes = "获取用户可用的优先级")

public ResponseBean<List<Integer>> getUserPriority(

@ApiParam(required = true, name = "userId", value = "用户id") @QueryParam("userId") String userId

) {

return securityGroupBusinessService.getUserPriority(userId);

}

@POST

@Path("/getSecurityGroupRole")

@ApiOperation(value = "查询安全组的规则", httpMethod = "POST", notes = "查询安全组的规则")

public ResponseBean<PageInfo<NetworkSecurityGroup>> getSecurityGroupRole(

@ApiParam(name = "page", value = "页号") @QueryParam(value = "page") Integer page,

@ApiParam(name = "pageNum", value = "每页几条") @QueryParam(value = "pageNum") Integer pageNum,

@ApiParam(required = true, name = "tagId", value = "安全组id") @QueryParam("tagId") String tagId,

@ApiParam(required = true, name = "direction", value = "出入方向") @QueryParam("direction") Integer direction

) {

return securityGroupBusinessService.getSecurityGroup(page, pageNum, tagId);

}

@POST

@Path("/getGroupList")

@ApiOperation(value = "查询安全组列表", httpMethod = "POST", notes = "查询安全组列表")

public ResponseBean<PageInfo<Map<String, Object>>> getGroupList(

@ApiParam(name = "page", value = "页号") @QueryParam(value = "page") Integer page,

@ApiParam(name = "pageNum", value = "每页几条") @QueryParam(value = "pageNum") Integer pageNum,

@ApiParam(name = "startTime", value = "开始时间") @QueryParam(value = "startTime") String startTime,

@ApiParam(name = "endTime", value = "结束时间") @QueryParam(value = "endTime") String endTime,

@ApiParam(name = "keyword", value = "搜索关键字") @QueryParam(value = "keyword") String keyword,

@ApiParam(name = "userId", value = "用户id") @QueryParam("userId") String userId,

@ApiParam(name = "mobile", value = "用户名") @QueryParam("mobile") String mobile

) {

return securityGroupBusinessService.getInGroup(page, pageNum, startTime, endTime, keyword, userId, mobile);

}

@POST

@Path("/getSecurityGroup")

@ApiOperation(value = "查询安全组详情", httpMethod = "POST", notes = "查询安全组详情")

public ResponseBean<NetworkSecurityGroupTag> getSecurityGroup(

@ApiParam(name = "id", value = "安全组id") @QueryParam(value = "id") String id

){

return securityGroupBusinessService.getSecurityGroup(id);

}

@POST

@Path("/updateSecurityGroup")

@ApiOperation(value = "编辑安全组", httpMethod = "POST", notes = "编辑安全组")

public ResponseBean<Void> updateSecurityGroup(

@ApiParam(name = "group", value = "安全组") @RequestParam NetworkSecurityGroupTag group

){

return securityGroupBusinessService.updateSecurityGroup(group);

}

@POST

@Path("/deleteSecurityGroup")

@ApiOperation(value = "删除安全组", httpMethod = "POST", notes = "删除安全组")

public ResponseBean<Void> deleteSecurityGroup(

@ApiParam(name = "id", value = "安全组id") @QueryParam(value = "id") String id

){

return securityGroupBusinessService.deleteSecurityGroup(id);

}

@POST

@Path("/deleteSecurityGroups")

@ApiOperation(value = "删除安全组", httpMethod = "POST", notes = "删除安全组")

public ResponseBean<Void> deleteSecurityGroup(

@ApiParam(required = true, name = "obj", value = "安全组参数") @RequestBody JSONObject obj

){

return securityGroupBusinessService.deleteSecurityGroup(obj);

}

@POST

@Path("/selectContainerIp")

@ApiOperation(value = "查询用户所有容器选择ip", httpMethod = "POST", notes = "查询用户所有容器选择ip")

public ResponseBean<List<CloudContainerAssets>> selectContainerIp(

@ApiParam(name = "serverCenterId", value = "服务中心id") @QueryParam(value = "serverCenterId") String serverCenterId,

@ApiParam(name = "userId", value = "用户id") @QueryParam(value = "userId") String userId,

@ApiParam(name = "name", value = "搜索名称") @QueryParam(value = "name") String name

){

return securityGroupBusinessService.selectContainerIp(serverCenterId, userId, name);

}

@POST

@Path("/selectClusterIp")

@ApiOperation(value = "查询用户所有集群选择ip", httpMethod = "POST", notes = "查询用户所有集群选择ip")

public ResponseBean<List<Cluster>> selectClusterIp(

@ApiParam(name = "serverCenterId", value = "服务中心id") @QueryParam(value = "serverCenterId") String serverCenterId,

@ApiParam(name = "userId", value = "用户id") @QueryParam(value = "userId") String userId,

@ApiParam(name = "name", value = "搜索名称") @QueryParam(value = "name") String name

){

return securityGroupBusinessService.selectClusterIp(serverCenterId, userId, name);

}

@POST

@Path("/selectAppGroupIp")

@ApiOperation(value = "查询用户的应用分组选择ip", httpMethod = "POST", notes = "查询用户的应用分组选择ip")

public ResponseBean<List<AppGroup>> selectAppGroupIp(

@ApiParam(name = "serverCenterId", value = "服务中心id") @QueryParam(value = "serverCenterId") String serverCenterId,

@ApiParam(name = "userId", value = "用户id") @QueryParam(value = "userId") String userId,

@ApiParam(name = "name", value = "搜索名称") @QueryParam(value = "name") String name

){

return securityGroupBusinessService.selectAppGroupIp(serverCenterId, userId, name);

}

@POST

@Path("/getIp")

@ApiOperation(value = "查询ip选取的容器、集群、应用分组的ip", httpMethod = "POST", notes = "查询ip选取的容器、集群、应用分组的ip")

public ResponseBean<List<String>> getIp(

@ApiParam(required = true, name = "obj", value = "安全组参数") @RequestBody JSONObject obj

){

return securityGroupBusinessService.getIp(obj);

}

@POST

@Path("/getUserContainerBindSecurity")

@ApiOperation(value = "绑定安全组选择容器", httpMethod = "POST", notes = "绑定安全组选择容器")

public ResponseBean<List<Map<String, Object>>> getUserContainerBindSecurity(

@ApiParam(name = "serverCenterId", value = "服务中心id") @QueryParam(value = "serverCenterId") String serverCenterId,

@ApiParam(name = "userId", value = "用户id") @QueryParam(value = "userId") String userId,

@ApiParam(name = "tagId", value = "安全组id") @QueryParam(value = "tagId") String tagId,

@ApiParam(name = "name", value = "名字搜索") @QueryParam(value = "name") String name

){

return securityGroupBusinessService.getUserContainerBindSecurity(serverCenterId, userId, tagId, name);

}

@POST

@Path("/getUserClusterBindSecurity")

@ApiOperation(value = "绑定安全组选择集群", httpMethod = "POST", notes = "绑定安全组选择集群")

public ResponseBean<List<Map<String, Object>>> getUserClusterBindSecurity(

@ApiParam(name = "serverCenterId", value = "服务中心id") @QueryParam(value = "serverCenterId") String serverCenterId,

@ApiParam(name = "userId", value = "用户id") @QueryParam(value = "userId") String userId,

@ApiParam(name = "tagId", value = "安全组id") @QueryParam(value = "tagId") String tagId,

@ApiParam(name = "name", value = "名字搜索") @QueryParam(value = "name") String name

){

return securityGroupBusinessService.getUserClusterBindSecurity(serverCenterId, userId, tagId, name);

}

@POST

@Path("/getUserAppGroupBindSecurity")

@ApiOperation(value = "绑定安全组选择应用分组", httpMethod = "POST", notes = "绑定安全组选择应用分组")

public ResponseBean<List<Map<String, Object>>> getUserAppGroupBindSecurity(

@ApiParam(name = "serverCenterId", value = "服务中心id") @QueryParam(value = "serverCenterId") String serverCenterId,

@ApiParam(name = "userId", value = "用户id") @QueryParam(value = "userId") String userId,

@ApiParam(name = "tagId", value = "安全组id") @QueryParam(value = "tagId") String tagId,

@ApiParam(name = "name", value = "名字搜索") @QueryParam(value = "name") String name

){

return securityGroupBusinessService.getUserAppGroupBindSecurity(serverCenterId, userId, tagId, name);

}

@POST

@Path("/securityGroupBindList")

@ApiOperation(value = "查询安全组绑定的成员带分页", httpMethod = "POST", notes = "查询安全组绑定的成员带分页")

public ResponseBean<PageInfo<Map<String, Object>>> securityGroupBindList(

@ApiParam(name = "page", value = "页号") @QueryParam(value = "page") Integer page,

@ApiParam(name = "pageNum", value = "每页几条") @QueryParam(value = "pageNum") Integer pageNum,

@ApiParam(required = true, name = "tagId", value = "安全组id") @QueryParam("tagId") String tagId

){

return securityGroupBusinessService.securityGroupBindList(page, pageNum, tagId);

}

}

package com.arp.service.rpcloud.network.resource;

import com.arp.common.rpcloud.commonapi.constants.exception\_code.ErrorCode;

import com.arp.common.rpcloud.commonapi.model.CloudContainerAssets;

import com.arp.common.rpcloud.remote.bean.ResponseBean;

import com.arp.common.rpcloud.remote.bean.network.BatchNetworkParams;

import com.arp.service.rpcloud.network.service.NetworkBusinessService;

import com.wordnik.swagger.annotations.Api;

import com.wordnik.swagger.annotations.ApiOperation;

import com.wordnik.swagger.annotations.ApiParam;

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

import org.springframework.stereotype.Component;

import javax.ws.rs.POST;

import javax.ws.rs.Path;

import javax.ws.rs.Produces;

import javax.ws.rs.QueryParam;

import javax.ws.rs.core.MediaType;

import java.util.List;

@Path("/businessNetwork")

@Api(value = "/businessNetwork", description = "网络服务", produces = MediaType.APPLICATION\_JSON)

@Produces(MediaType.APPLICATION\_JSON)

@Component

public class NetworkBusinessResource {

@Autowired

private NetworkBusinessService networkBusinessService;

@POST

@Path("/getPatchNetworkParams")

@ApiOperation(value = "查询网关容器列表",httpMethod = "POST", notes = "查询网关容器列表")

public ResponseBean<List<BatchNetworkParams>> getPatchNetworkParams(

@ApiParam(name = "containerId", value = "容器主键") @QueryParam("containerId") String containerId

){

//ResponseBean<List<Map<String, Object>>> result = this.networkBusinessService.getGatewayList(userId);

List<BatchNetworkParams> patchNetworkParams = this.networkBusinessService.getPatchNetworkParams(containerId);

ResponseBean<List<BatchNetworkParams>> responseBean = new ResponseBean<>();

responseBean.setCodeAndMsg(ErrorCode.SUCCESS\_code.getCode(), ErrorCode.SUCCESS\_code.getMsg());

responseBean.setData(patchNetworkParams);

return responseBean;

}

@POST

@Path("/getShellBoxUrl")

@ApiOperation(value = "获取shellbox的请求路径",httpMethod = "POST", notes = "获取shellbox的请求路径")

public ResponseBean<String> getShellBoxUrl(

@ApiParam(name = "containerId", value = "容器主键") @QueryParam("containerId") String containerId

){

return networkBusinessService.getShellBoxUrl(containerId);

}

@POST

@Path("/getWebconsuleUrl")

@ApiOperation(value = "获取ip地址",httpMethod = "POST", notes = "获取ip地址")

public ResponseBean<String> getWebconsuleUrl(

@ApiParam(name = "containerId", value = "容器主键") @QueryParam("containerId") String containerId

){

return networkBusinessService.getWebconsuleUrl(containerId);

}

@POST

@Path("/getEnAddr")

@ApiOperation(value = "获取enaddr",httpMethod = "POST", notes = "获取enaddr")

public ResponseBean<String> getEnAddr(

@ApiParam(name = "vmaddr", value = "vmaddr -- ip:prot") @QueryParam("vmaddr") String vmaddr,

@ApiParam(name = "userId", value = "userId") @QueryParam("userId") String userId

){

return networkBusinessService.getEnAddr(vmaddr,userId);

}

@POST

@Path("/getUserGateway")

@ApiOperation(value = "网关列表（不包含ip等信息）",httpMethod = "POST", notes = "网关列表（不包含ip等信息）")

public ResponseBean<List<CloudContainerAssets>> getUserGateway(

@ApiParam(name = "userId", value = "用户id") @QueryParam("userId") String userId,

@ApiParam(name = "serverCenterId", value = "用户中心id") @QueryParam("serverCenterId") String serverCenterId

){

return networkBusinessService.getUserGateway(userId, serverCenterId);

}

}

package com.arp.service.rpcloud.network.resource;

import java.util.List;

import java.util.Map;

import javax.ws.rs.POST;

import javax.ws.rs.Path;

import javax.ws.rs.Produces;

import javax.ws.rs.QueryParam;

import javax.ws.rs.core.MediaType;

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

import org.springframework.stereotype.Component;

import org.springframework.web.bind.annotation.RequestBody;

import com.alibaba.fastjson.JSONObject;

import com.github.pagehelper.PageInfo;

import com.arp.common.rpcloud.remote.bean.ResponseBean;

import com.arp.service.rpcloud.network.service.GatewayNetworkService;

import com.wordnik.swagger.annotations.Api;

import com.wordnik.swagger.annotations.ApiOperation;

import com.wordnik.swagger.annotations.ApiParam;

@Path("/gatewayNetwork")

@Api(value = "/gatewayNetwork", description = "网关联通服务", produces = MediaType.APPLICATION\_JSON)

@Produces(MediaType.APPLICATION\_JSON)

@Component

public class GatewayNetworkResource {

@Autowired

private GatewayNetworkService gatewayNetworkService;

@POST

@Path("/connectGateway")

@ApiOperation(value = "关联网关",httpMethod = "POST", notes = "关联网关")

public ResponseBean<Void> connectGateway(

@ApiParam(required = true, name = "params", value = "添加参数") @RequestBody JSONObject obj

){

return this.gatewayNetworkService.connectGateway(obj);

}

@POST

@Path("/getConnectGatewayList")

@ApiOperation(value = "关联网关选择列表",httpMethod = "POST", notes = "关联网关选择列表")

public ResponseBean<List<Map<String, Object>>> getConnectGatewayList(

@ApiParam(name = "gatewayId", value = "网关id") @QueryParam("gatewayId") String gatewayId,

@ApiParam(name = "userId", value = "用户id") @QueryParam("userId") String userId,

@ApiParam(name = "serverCenterId", value = "用户中心") @QueryParam("serverCenterId") String serverCenterId

){

return this.gatewayNetworkService.getConnectGatewayList(gatewayId, userId, serverCenterId);

}

@POST

@Path("/disConnectGateway")

@ApiOperation(value = "取消关联网关",httpMethod = "POST", notes = "取消关联网关")

public ResponseBean<Void> disConnectGateway(

@ApiParam(required = true, name = "params", value = "添加参数") @RequestBody JSONObject obj

){

return this.gatewayNetworkService.disConnectGateway(obj);

}

@POST

@Path("/getGatewayInfo")

@ApiOperation(value = "网关详情",httpMethod = "POST", notes = "网关详情")

public ResponseBean<Map<String, Object>> getGatewayInfo(

@ApiParam(required = true, name = "id", value = "网关id") @QueryParam("id") String id

){

return this.gatewayNetworkService.getGatewayInfo(id);

}

@POST

@Path("/getGatewayConnectList")

@ApiOperation(value = "查询网关详情打通路由器列表",httpMethod = "POST", notes = "查询网关详情打通路由器列表")

public ResponseBean<PageInfo<Map<String, Object>>> getGatewayConnectList(

@ApiParam(required = true, name = "page", value = "页号") @QueryParam(value = "page") Integer page,

@ApiParam(required = true, name = "pageNum", value = "每页几条") @QueryParam(value = "pageNum") Integer pageNum,

@ApiParam(required = true, name = "gatewayId", value = "网关id") @QueryParam("gatewayId") String gatewayId

){

return this.gatewayNetworkService.getGatewayConnectList(page, pageNum, gatewayId);

}

@POST

@Path("/gatewayConnectActive")

@ApiOperation(value = "取消关联网关",httpMethod = "POST", notes = "取消关联网关")

public void gatewayConnectActive(

@ApiParam(required = true, name = "params", value = "添加参数") @RequestBody JSONObject obj

){

this.gatewayNetworkService.gatewayConnectActive(obj);

};

}

package com.arp.service.rpcloud.network.mapper;

import org.apache.ibatis.annotations.Select;

import org.apache.ibatis.annotations.SelectProvider;

import org.springframework.stereotype.Repository;

import java.util.List;

import java.util.Map;

@Repository

public interface SecurityGroupBusinessMapper {

@Select("select distinct a.priority from network\_security\_group a left join network\_security\_group\_tag b on a.group\_tag\_id = b.id where b.user\_id = #{userId}")

public List<Integer> getUserPriority(String userId);

// @Select("select c.\*,b.webname,b.id as container\_id from cloud\_network\_device a left join cloud\_container\_assets b on a.cloud\_assets\_id = b.id left join public\_ip c on a.ip\_id = c.id where a.status = 0 and a.type = '1' and b.user\_id = #{userId} and c.id in (select ip\_id from network\_securitygroup\_ip where type = 1 and tag\_id = #{tagId})")

// public List<Map<String, Object>> getUserGroupBindPublicIp(Map<String, Object> param);

//

// @Select("select c.\*,b.webname from cloud\_network\_device a left join cloud\_container\_assets b on a.cloud\_assets\_id = b.id left join public\_ip c on a.ip\_id = c.id where a.status = 0 and a.type = '1' and b.is\_gateway = 0 and b.user\_id = #{userId} and c.id not in (select ip\_id from network\_securitygroup\_ip where type = 1)")

// public List<Map<String, Object>> getUserGroupUnBindPublicIp(String userId);

//

// @Select("select c.\*,b.webname,b.id as container\_id from cloud\_network\_device a left join cloud\_container\_assets b on a.cloud\_assets\_id = b.id left join ip c on a.ip\_id = c.id where a.status = 0 and a.type = '0' and b.user\_id = #{userId} and c.id in (select ip\_id from network\_securitygroup\_ip where type = 0 and tag\_id = #{tagId})")

// public List<Map<String, Object>> getUserGroupBindIp(Map<String, Object> param);

//

// @Select("select c.\*,b.webname from cloud\_network\_device a left join cloud\_container\_assets b on a.cloud\_assets\_id = b.id left join ip c on a.ip\_id = c.id where a.status = 0 and a.type = '0' and b.is\_gateway = 0 and b.user\_id = #{userId} and c.id not in (select ip\_id from network\_securitygroup\_ip where type = 0)")

// public List<Map<String, Object>> getUserGroupUnBindIp(String userId);

// @Select("select a.\* from network\_security\_group\_tag a where a.id in (select tag\_id from network\_securitygroup\_ip where ip\_id = #{ipId} and type = #{type})")

// public List<Map<String, Object>> getIpBindSecurityGroup(Integer ipId, Integer type);

@SelectProvider(type=SecurityGroupBusinessProvider.class, method="getInGroup")

public List<Map<String, Object>> getInGroup(Map<String, Object> param);

@Select("select b.p\_id,b.id\_type from slot a left join homologous\_relation\_saci b on a.id = b.slot\_id where a.id in (SELECT slot\_id from `app\_slot\_relation` where app\_id = #{appId})")

public List<Map<String, Object>> getAppSlot(String appId);

@SelectProvider(type=SecurityGroupBusinessProvider.class, method="getUserContainerBindSecurity")

public List<Map<String, Object>> getUserContainerBindSecurity(Map<String, Object> param);

@SelectProvider(type=SecurityGroupBusinessProvider.class, method="getUserClusterBindSecurity")

public List<Map<String, Object>> getUserClusterBindSecurity(Map<String, Object> param);

@SelectProvider(type=SecurityGroupBusinessProvider.class, method="getUserAppGroupBindSecurity")

public List<Map<String, Object>> getUserAppGroupBindSecurity(Map<String, Object> param);

@Select("select a.\*,b.webname as name " +

" from `network\_security\_tag\_relation` a left join cloud\_container\_assets b on a.target\_id = b.id where a.type = 3 and a.tag\_id = #{tagId}" +

"union " +

"select a.\*,b.cluster\_name as name " +

" from `network\_security\_tag\_relation` a left join cluster b on a.target\_id = b.id where a.type = 2 and a.tag\_id = #{tagId}" +

"union " +

"select a.\*,b.`app\_group\_name` as name " +

" from `network\_security\_tag\_relation` a left join app\_group b on a.target\_id = b.id where a.type = 1 and a.tag\_id = #{tagId}")

public List<Map<String, Object>> securityGroupBindList(String tagId);

}

package com.arp.service.rpcloud.network.mapper;

import java.util.Map;

public class SecurityGroupBusinessProvider {

/\*\*

\* 安全组列表sql

\* @param param

\* @return

\*/

public String getInGroup(Map<String, Object> param) {

StringBuffer sql = new StringBuffer();

sql.append("select aaa.\*,ifnull(bbb.c,0) c, u.mobile from network\_security\_group\_tag aaa left join ");

sql.append("( ");

sql.append("select tag\_id, COUNT(\*) c from `network\_security\_tag\_relation` group by tag\_id");

sql.append(") bbb on aaa.id = bbb.tag\_id left join user\_info u on aaa.user\_id = u.user\_id where 1 = 1 ");

if (param.get("userId") != null){

sql.append(" and aaa.user\_id = #{userId} ");

}

if (param.get("mobile") != null){

sql.append(" and u.mobile = #{mobile} ");

}

if (param.get("startTime") != null){

sql.append("and aaa.create\_time >= date\_format(#{startTime},'%Y-%c-%d %H:%i:%s') ");

}

if (param.get("endTime") != null){

sql.append("and aaa.create\_time <= date\_format(#{endTime},'%Y-%c-%d %H:%i:%s') ");

}

if (param.get("keyword") != null){

sql.append("and (aaa.id like concat('%', #{keyword}, '%') or aaa.tagName like concat('%', #{keyword}, '%') ) ");

}

sql.append(" order by aaa.create\_time desc");

return sql.toString();

}

public String getUserContainerBindSecurity(Map<String, Object> param) {

StringBuffer sb = new StringBuffer();

sb.append("select id,webname from cloud\_container\_assets where `status` in (1, 2) and is\_delete not in (1,2) and is\_gateway = 0 and is\_display = 1 and server\_center\_id = #{serverCenterId} and user\_id = #{userId} and id not in (select target\_id from `network\_security\_tag\_relation` where type = 3 and tag\_id = #{tagId})");

if (param.get("name") != null) {

sb.append("and webname like concat('%', #{name}, '%') ");

}

return sb.toString();

}

public String getUserClusterBindSecurity(Map<String, Object> param) {

StringBuffer sb = new StringBuffer();

sb.append("select id,cluster\_name from cluster where status in (1,2) and is\_delete = 0 and user\_id = #{userId} and server\_center\_id = #{serverCenterId} and id not in (select target\_id from `network\_security\_tag\_relation` where type = 2 and tag\_id = #{tagId})");

if (param.get("name") != null) {

sb.append("and cluster\_name like concat('%', #{name}, '%') ");

}

return sb.toString();

}

public String getUserAppGroupBindSecurity(Map<String, Object> param) {

StringBuffer sb = new StringBuffer();

sb.append("select id,app\_group\_name from app\_group where user\_id = #{userId} and is\_delete = 0 and server\_center\_id = #{serverCenterId} and id not in (select target\_id from `network\_security\_tag\_relation` where type = 1 and tag\_id = #{tagId}) ");

if (param.get("name") != null) {

sb.append("and app\_group\_name like concat('%', #{name}, '%') ");

}

return sb.toString();

}

}

package com.arp.service.rpcloud.network.mapper;

import com.arp.common.rpcloud.commonapi.model.NetworkContainerMac;

import org.apache.ibatis.annotations.Result;

import org.apache.ibatis.annotations.Results;

import org.apache.ibatis.annotations.Select;

import org.apache.ibatis.type.JdbcType;

import org.springframework.stereotype.Repository;

import java.util.List;

import java.util.Map;

/\*\*

\* Created by zhoujia on 2017/3/20.

\* 自定义mapper

\*/

@Repository

public interface IpBusinessMapper {

/\*\*

\* 通过id查找ip信息

\* @param param ipId

\* @return

\*/

@Select({

"SELECT ip.id,ip.ip,ip.mask,ip.gateway,vlan.vlan\_name ,ip.mask from ip \n" +

"LEFT JOIN intr\_ip\_para ipp ON ip.ip\_para\_id = ipp.id \n" +

"LEFT JOIN vlan ON vlan.id = ipp.vlan\_id \n" +

"where ip.id = #{ipId} "

})

List<Map<String, Object>> findIpInfo(Map<String, Object> param);

@Select({

"SELECT ip.\* from cloud\_container\_assets cca \n" +

"left JOIN cloud\_network\_device cnd on cnd.cloud\_assets\_id = cca.id\n " +

"left join ip on ip.id = cnd.ip\_id\n " +

"where cca.container\_id = #{containerId} and cnd.`status`=0 "

})

List<Map<String, Object>> findIpInfoByContainerId(Map<String, Object> param);

/\*\*

\* 通过id查找公网ip信息

\* @param ipId

\* @return

\*/

@Select({

"SELECT pubip.gateway, pubip.ip from public\_ip pubip left join public\_ip\_info pii on pubip.id = pii.public\_ip\_id\n" +

"where pubip.id = #{ipId} and NOW() < pii.end\_time "

})

List<Map<String, Object>> findPubIpInfoNotEndTime(String ipId);

@Select("select id,mac,status,create\_time,update\_time from network\_container\_mac where status=0 order by id limit 0,1")

@Results({

@Result(column = "id", property = "id", jdbcType = JdbcType.INTEGER, id = true),

@Result(column = "mac", property = "mac", jdbcType = JdbcType.VARCHAR),

@Result(column = "status", property = "status", jdbcType = JdbcType.INTEGER),

@Result(column = "update\_time", property = "updateTime", jdbcType = JdbcType.TIMESTAMP),

@Result(column = "create\_time", property = "createTime", jdbcType = JdbcType.TIMESTAMP)

})

NetworkContainerMac findNoUsedMac();

}

package com.arp.service.rpcloud.network.util;

import java.util.ArrayList;

import java.util.List;

import com.arp.common.rpcloud.commonapi.util.IpUtil;

/\*\*

\* Created by zhoujia on 2017/5/16.

\* 判断两个ip是否是统一个网段

\*/

public class IpUtils {

public static boolean isSame(String ip1,String ip2,String mask) {

//255.255.255.0

//192.168.1.1

int i1 = get2Num(ip1);

int i2 = get2Num(ip2);

int maskInt = get2Num(mask);

//int i3 = i1&i2;

//i1&maskInt;

//System.out.println(i1 + " " + i2 + " " + i3);

return (i1&maskInt) == (i2&maskInt) ;

}

private static int get2Num(String mask) {

String[] str = mask.split("\\.");

String integer1 = "";

int t = 0;

for (int i=0;i<str.length;i++) {

integer1 = Integer.toBinaryString(Integer.parseInt(str[i]));

Integer integer = Integer.valueOf(integer1, 2);

int num = integer << 8\*(3-i);

t = t + num;

}

return t;

}

public static List<String> splitIp(String ip){

List<String> result = new ArrayList<>();

String[] ips = ip.split(",");

for (String spIp : ips) {

if (spIp.contains("-")) {

List<String> ipMaskList = IpUtil.ipToMask(spIp);

result.addAll(ipMaskList);

}else {

result.add(spIp);

}

}

return result;

}

}

package com.arp.service.rpcloud.network.util;

/\*\*

\* =======================================================

\*

\* @Version ：0.0.1

\* @Description :网络下发常规类型

\* ========================================================

\*/

public enum NetworkDlType {

TCP("tcp"),

UDP("udp"),

ICMP("icmp");

//私有成员变量，保存名称

private String value;

public String getValue() {

return value;

}

//带参构造函数

NetworkDlType(String value) {

this.value = value;

}

public static String calculate(Integer protocolType) {

switch (protocolType) {

case 0:

return NetworkDlType.TCP.getValue();

case 1:

return NetworkDlType.UDP.getValue();

case 2:

return NetworkDlType.ICMP.getValue();

default:

throw new AssertionError("Unknown operations " + protocolType);

}

}

}