package cn.g4b.fm.service.szg;

import cn.g4b.fm.common.util.HttpUtil;

import cn.g4b.fm.common.util.LogUtils;

import cn.g4b.fm.common.util.StringUtils;

import cn.g4b.fm.dao.sys.CbAppMapper;

import cn.g4b.fm.dao.szg.ServiceProviderMapper;

import cn.g4b.fm.model.sys.CbApp;

import cn.g4b.fm.model.szg.\*;

import com.fasterxml.jackson.databind.ObjectMapper;

import org.slf4j.Logger;

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

import org.springframework.stereotype.Service;

import java.util.Arrays;

import java.util.HashMap;

import java.util.Map;

/\*\*

\* 应用通知服务

\*/

@Service

public class AppNoticService {

private Logger logger = LogUtils.getPlatformLogger();

@Autowired

private CbAppMapper appMapper;

@Autowired

private ServiceProviderMapper serviceProviderMapper;

public void certNotice(SzgCertOrder order){

Thread noticeThread = new Thread() {

@Override

public void run() {

//通知第三方

if(StringUtils.isEmpty(order.getAppId()))

return;

CbApp cbApp = appMapper.selectByPrimaryKey(order.getAppId());

if(cbApp == null || StringUtils.isEmpty(cbApp.getCallbackUrl()))

return;

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

notMap.put("orderResult",order.getOrderStatus().intValue()==3?1:0);

notMap.put("orderId",order.getOrderId());

notMap.put("msg",order.getMsg());

notMap.put("noticType","cert");

notMap.put("time",System.currentTimeMillis());

ObjectMapper objectMapper = new ObjectMapper();

try{

String body = objectMapper.writeValueAsString(notMap);

logger.info("证书生成结果通知：orderId===={} 回调url===={}",order.getOrderId(),cbApp.getCallbackUrl());

String s = HttpUtil.sendHttpPost(cbApp.getCallbackUrl(), body, null);

logger.info("返回结果====>>{}",s);

}catch (Exception e){

e.printStackTrace();

}

}

};

noticeThread.start();

}

public void signNotice(SzgSignOrder order){

Thread noticeThread = new Thread() {

@Override

public void run() {

//通知第三方

if(StringUtils.isEmpty(order.getAppId()))

return;

CbApp cbApp = appMapper.selectByPrimaryKey(order.getAppId());

if(cbApp == null || StringUtils.isEmpty(cbApp.getCallbackUrl()))

return;

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

notMap.put("orderResult",order.getOrderStatus().intValue()==3?1:0);

notMap.put("orderId",order.getOrderId());

notMap.put("msg",order.getMsg());

notMap.put("noticType","sign");

notMap.put("time",System.currentTimeMillis());

notMap.put("signValue",order.getSignValue());

ObjectMapper objectMapper = new ObjectMapper();

try{

String body = objectMapper.writeValueAsString(notMap);

logger.info("证书签名结果通知：orderId===={} 回调url===={}",order.getOrderId(),cbApp.getCallbackUrl());

String s = HttpUtil.sendHttpPost(cbApp.getCallbackUrl(), body, null);

logger.info("返回结果====>>{}",s);

}catch (Exception e){

e.printStackTrace();

}

}

};

noticeThread.start();

}

public void sealNotice(SzgSealOrder order,String msg){

Thread noticeThread = new Thread() {

@Override

public void run() {

//通知第三方

if(StringUtils.isEmpty(order.getAppId()))

return;

CbApp cbApp = appMapper.selectByPrimaryKey(order.getAppId());

if(cbApp == null || StringUtils.isEmpty(cbApp.getCallbackUrl()))

return;

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

notMap.put("orderResult",order.getOrderStatus().intValue()==3?1:0);

notMap.put("orderId",order.getOrderId());

notMap.put("msg",msg);

notMap.put("noticType","seal");

notMap.put("time",System.currentTimeMillis());

ObjectMapper objectMapper = new ObjectMapper();

try{

String body = objectMapper.writeValueAsString(notMap);

logger.info("印章生成结果通知：orderId===={} 回调url===={}",order.getOrderId(),cbApp.getCallbackUrl());

String s = HttpUtil.sendHttpPost(cbApp.getCallbackUrl(), body, null);

logger.info("返回结果====>>{}",s);

}catch (Exception e){

e.printStackTrace();

}

}

};

noticeThread.start();

}

public void stampNotice(SzgStampOrder order, String msg){

Thread noticeThread = new Thread() {

@Override

public void run() {

//通知第三方

if(StringUtils.isEmpty(order.getAppId()))

return;

CbApp cbApp = appMapper.selectByPrimaryKey(order.getAppId());

if(cbApp == null || StringUtils.isEmpty(cbApp.getCallbackUrl()))

return;

Map noticInfo = new HashMap();

noticInfo.put("orderId", order.getOrderId());

noticInfo.put("orderResult", Integer.valueOf((order.getOrderStatus().intValue() == 3) ? 1 : 0));

noticInfo.put("msg", order.getMsg());

noticInfo.put("time", System.currentTimeMillis());

noticInfo.put("noticType", "stamp");

noticInfo.put("stampingSign", order.getSealSign());

ObjectMapper objectMapper = new ObjectMapper();

try{

String body = objectMapper.writeValueAsString(noticInfo);

logger.info("盖章生成结果通知：orderId===={} 回调url===={}",order.getOrderId(),cbApp.getCallbackUrl());

logger.info("通知信息======>>>{}",body);

String s = HttpUtil.sendHttpPost(cbApp.getCallbackUrl(), body, null);

logger.info("返回结果====>>{}",s);

}catch (Exception e){

e.printStackTrace();

}

}

};

noticeThread.start();

}

/\*\*

\* 新印章订单回调CA服务商

\* @param orderId

\* @param noticeType applyNewSeal 新印章

\* applySealChange 印章变更

\* applyStamping 盖章订单

\* applyNewCert 新证书

\* applySign 签名

\*/

public void newSealNotice(String orderId,String spNo,String noticeType) {

Thread noticeThread = new Thread() {

@Override

public void run() {

//通知第三方

if(StringUtils.isEmpty(orderId) || StringUtils.isEmpty(noticeType))

return;

ServiceProvider serviceProvider = serviceProviderMapper.getServiceProviderBySpNo(spNo);

if(serviceProvider == null || StringUtils.isEmpty(serviceProvider.getCallbackUrl()))

return;

Map noticInfo = new HashMap();

noticInfo.put("orderIds", Arrays.asList(orderId));

noticInfo.put("time", System.currentTimeMillis());

noticInfo.put("noticType", noticeType);

ObjectMapper objectMapper = new ObjectMapper();

try{

String body = objectMapper.writeValueAsString(noticInfo);

logger.info("盖章生成结果通知：orderId===={} 回调url===={}",orderId,serviceProvider.getCallbackUrl());

logger.info("通知信息======>>>{}",body);

String s = HttpUtil.sendHttpPost(serviceProvider.getCallbackUrl(), body, null);

logger.info("返回结果====>>{}",s);

}catch (Exception e){

e.printStackTrace();

}

}

};

noticeThread.start();

}

}

package cn.g4b.fm.service.szg;

import cn.g4b.fm.common.util.LogUtils;

import cn.g4b.fm.dao.szg.SzgSealOrderMapper;

import cn.g4b.fm.dao.szg.SzgStampOrderMapper;

import org.slf4j.Logger;

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

import org.springframework.stereotype.Service;

import java.text.NumberFormat;

import java.text.SimpleDateFormat;

import java.util.\*;

/\*\*

\* Created by Asus on 2021/3/17.

\*/

@Service

public class AppOrderShowService {

private Logger logger = LogUtils.getPlatformLogger();

@Autowired

private SzgSealOrderMapper szgSealOrderMapper;

@Autowired

private SzgStampOrderMapper szgStampOrderMapper;

public Map workOrderByCompany(String spNo,String appId){

String startTime = null;

String nowEndTime = null;

SimpleDateFormat format = new SimpleDateFormat("yyyy-MM-dd");

Calendar c = Calendar.getInstance();

c.setTime(new Date());

Date nowStart = c.getTime();

startTime = format.format(nowStart);

logger.info("workOrderByCompany day startTime:" + startTime );

c.add(Calendar.DATE, + 1);

Date d = c.getTime();

nowEndTime = format.format(d);

List<Map<String, Object>> list = szgSealOrderMapper.workOrderByCompany(spNo,appId,startTime, nowEndTime);

// Map<String,Long> item = new LinkedHashMap<>();

// item.put(startTime,0l);

Long nowData = 0l;

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

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

java.sql.Date date = (java.sql.Date) ma.get("day");

// String dayth = format.format(date);

Long count = (Long) ma.get("countNum");

if(count!=0l){

nowData = count;

}

}

}

Map restMap = new LinkedHashMap();

restMap.put("nowData",nowData);

c.add(Calendar.DATE, - 7);

Date d1 = c.getTime();

String startWTime = format.format(d1);

logger.info(" SealOrder weeklist startWTime:" + startWTime );

List<Map<String, Object>> weeklist = szgSealOrderMapper.workOrderByCompany(spNo,appId,startWTime, nowEndTime);

c.add(Calendar.DATE, -1);

Date d2 = c.getTime();

c.add(Calendar.DATE, +7);

Date d3 = c.getTime();

Calendar dd = Calendar.getInstance();//定义日期实例

dd.setTime(d2);//设置日期起始时间

Map<String,Long> item1 = new LinkedHashMap<>();

while (dd.getTime().before(d3)) {//判断是否到结束日期

dd.add(Calendar.DATE, 1);//进行当前日期加1

String str = format.format(dd.getTime());

item1.put(str,0l);

}

long weekTotal = 0l;

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

for(Map<String,Object> ma:weeklist){

java.sql.Date date = (java.sql.Date) ma.get("day");

String weekth = format.format(date);

Long count = (Long) ma.get("countNum");

Long weekcount = (Long) ma.get("countNum");

weekTotal += weekcount;

if(count!=0l){

item1.put(weekth,count);

}

}

}

logger.info("SealOrder weeklist weekTotal:" + weekTotal);

restMap.put("weekData",weekTotal);

restMap.put("weekList",item1);

logger.info("SealOrder restMap:" + restMap );

return restMap;

}

public Map workOrderByPerson(String spNo,String appId){

String startTime = null;

String nowEndTime = null;

SimpleDateFormat format = new SimpleDateFormat("yyyy-MM-dd");

Calendar c = Calendar.getInstance();

c.setTime(new Date());

Date nowStart = c.getTime();

startTime = format.format(nowStart);

logger.info("workOrderByPerson day startTime:" + startTime );

c.add(Calendar.DATE, + 1);

Date d = c.getTime();

nowEndTime = format.format(d);

List<Map<String, Object>> list = szgSealOrderMapper.workOrderByPer(spNo,appId,startTime, nowEndTime);

// Map<String,Long> item = new LinkedHashMap<>();

// item.put(startTime,0l);

Long nowPerData = 0l;

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

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

java.sql.Date date = (java.sql.Date) ma.get("day");

// String dayth = format.format(date);

Long count = (Long) ma.get("countNum");

if(count!=0l){

nowPerData = count;

}

}

}

Map restMap = new LinkedHashMap();

restMap.put("nowPerData",nowPerData);

c.add(Calendar.DATE, - 7);

Date d1 = c.getTime();

String startWTime = format.format(d1);

logger.info(" SealOrder weeklist startWTime:" + startWTime );

List<Map<String, Object>> weeklist = szgSealOrderMapper.workOrderByPer(spNo,appId,startWTime, nowEndTime);

c.add(Calendar.DATE, -1);

Date d2 = c.getTime();

c.add(Calendar.DATE, +7);

Date d3 = c.getTime();

Calendar dd = Calendar.getInstance();//定义日期实例

dd.setTime(d2);//设置日期起始时间

Map<String,Long> item1 = new LinkedHashMap<>();

while (dd.getTime().before(d3)) {//判断是否到结束日期

dd.add(Calendar.DATE, 1);//进行当前日期加1

String str = format.format(dd.getTime());

item1.put(str,0l);

}

long weekPerTotal = 0l;

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

for(Map<String,Object> ma:weeklist){

java.sql.Date date = (java.sql.Date) ma.get("day");

String weekth = format.format(date);

Long count = (Long) ma.get("countNum");

Long weekcount = (Long) ma.get("countNum");

weekPerTotal += weekcount;

if(count!=0l){

item1.put(weekth,count);

}

}

}

logger.info("SealOrder weeklist weekPerTotal:" + weekPerTotal);

restMap.put("weekPerData",weekPerTotal);

restMap.put("weekPerList",item1);

logger.info("SealOrder restMap:" + restMap );

return restMap;

}

public Map workOrder(String spNo,String appId){

Map restMap = new LinkedHashMap();

String startTime = null;

String nowEndTime = null;

SimpleDateFormat format = new SimpleDateFormat("yyyy-MM-dd");

Calendar c = Calendar.getInstance();

c.setTime(new Date());

c.add(Calendar.DATE, + 1);

Date nowStart = c.getTime();

nowEndTime = format.format(nowStart);

logger.info("SealOrder nowEndTime:" + nowEndTime );

try {

c.add(Calendar.DATE, - 7);

Date d = c.getTime();

startTime = format.format(d);

logger.info(" SealOrder startTime:" + startTime );

List<Map<String, Object>> sealList = szgSealOrderMapper.sealCountByMonth(startTime, nowEndTime,spNo,appId);

List<Map<String, Object>> stampList = szgStampOrderMapper.stampCountByMonth(startTime, nowEndTime,spNo,appId);

List<Map> sealOrder = szgSealOrderMapper.sealTotalAndCompleted(startTime, nowEndTime,spNo,appId);

List<Map> stampOrder = szgStampOrderMapper.stampTotalAndCompleted(startTime, nowEndTime,spNo,appId);

c.add(Calendar.DATE, -1);

Date d1 = c.getTime();

c.add(Calendar.DATE, +7);

Date d2 = c.getTime();

Calendar dd = Calendar.getInstance();//定义日期实例

dd.setTime(d1);//设置日期起始时间

Map<String,Long> sealitem = new LinkedHashMap<>();

Map<String,Long> stampitem = new LinkedHashMap<>();

while (dd.getTime().before(d2)) {//判断是否到结束日期

dd.add(Calendar.DATE, 1);//进行当前日期加1

String str = format.format(dd.getTime());

sealitem.put(str,0l);

stampitem.put(str,0l);

}

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

for(Map<String,Object> seal:sealList){

java.sql.Date date = (java.sql.Date) seal.get("day");

String weekth = format.format(date);

Long count = (Long) seal.get("countNum");

if(count!=0l){

sealitem.put(weekth,count);

}

}

}

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

for(Map<String,Object> stamp:stampList){

java.sql.Date date = (java.sql.Date) stamp.get("day");

String weekth = format.format(date);

Long count = (Long) stamp.get("countNum");

if(count!=0l){

stampitem.put(weekth,count);

}

}

}

sealitem.forEach((key,value) -> stampitem.merge(key,value,Long::sum));

restMap.put("weekOrderMap",stampitem);

Long sealTotalNum = 0l;

Long sealCompletedNum = 0l;

Long stampTotalNum = 0l;

Long stampCompletedNum = 0l;

logger.info("====>>>sealOrder"+sealOrder);

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

for (Map seal : sealOrder){

if(seal.get("group").equals("get")){

sealTotalNum = (Long) seal.get("countNum");

}else if(seal.get("group").equals("completed")){

sealCompletedNum = (Long) seal.get("countNum");

}

}

}

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

for (Map stamp : stampOrder){

if(stamp.get("group").equals("get")){

stampTotalNum = (Long) stamp.get("countNum");

}else if(stamp.get("group").equals("completed")){

stampCompletedNum = (Long) stamp.get("countNum");

}

}

}

Long total = sealTotalNum + stampTotalNum;

Long completed = sealCompletedNum + stampCompletedNum;

restMap.put("weekTotal",total);

restMap.put("weekCompleted",completed);

NumberFormat numberFormat = NumberFormat.getInstance();

// 设置精确到小数点后2位

numberFormat.setMaximumFractionDigits(2);

if(total.intValue()!=0){

restMap.put("weekCompPercentage",numberFormat.format(Float.parseFloat(String.valueOf(completed)) / Float.parseFloat(String.valueOf(total)) \* 100)+"%");

}else{

restMap.put("weekCompPercentage","0.00%");

}

logger.info("workOrder restMap:" + restMap );

}catch (Exception e){

e.printStackTrace();

}

return restMap;

}

}

package cn.g4b.fm.service.szg;

import cn.g4b.fm.common.model.PageBean;

import cn.g4b.fm.common.util.LogUtils;

import cn.g4b.fm.dao.sys.CbAppMapper;

import cn.g4b.fm.dao.szg.ServiceProviderMapper;

import cn.g4b.fm.dao.szg.SzgSealOrderMapper;

import cn.g4b.fm.dao.szg.SzgStampOrderMapper;

import cn.g4b.fm.model.szg.SzgSealOrderVo;

import cn.g4b.fm.model.szg.SzgStampOrderVo;

import com.github.pagehelper.Page;

import com.github.pagehelper.PageHelper;

import org.slf4j.Logger;

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

import org.springframework.stereotype.Service;

import java.text.NumberFormat;

import java.util.ArrayList;

import java.util.HashMap;

import java.util.List;

import java.util.Map;

/\*\*

\* Created by Asus on 2021/2/5.

\*/

@Service

public class GeneralCountService {

private Logger logger = LogUtils.getPlatformLogger();

@Autowired

private ServiceProviderMapper serviceProviderMapper;

@Autowired

private SzgSealOrderMapper szgSealOrderMapper;

@Autowired

private SzgStampOrderMapper szgStampOrderMapper;

@Autowired

private CbAppMapper cbAppMapper;

public PageBean findProviderOrderList(PageBean pageBean, SzgSealOrderVo szgSealOrderVo){

if(pageBean==null)

pageBean = new PageBean();

Page page = PageHelper.startPage(pageBean.getPage(),pageBean.getPageSize());

List<Map> providerList = serviceProviderMapper.getServiceProviderList(null,null);

List<Map> sealCtInfo = szgSealOrderMapper.getSealCtInfo(szgSealOrderVo);

SzgStampOrderVo szgStampOrderVo = new SzgStampOrderVo();

szgStampOrderVo.setStartTime(szgSealOrderVo.getStartTime());

szgStampOrderVo.setEndTime(szgSealOrderVo.getEndTime());

List<Map> stampCtInfo = szgStampOrderMapper.getStampCtInfo(szgStampOrderVo);

List orderCountVoList = new ArrayList();

if((sealCtInfo!=null && !sealCtInfo.isEmpty())||(stampCtInfo!=null && !stampCtInfo.isEmpty())){

for (Map m : providerList){

Map mr = new HashMap();

mr.put("spNo",m.get("spNo"));

mr.put("spName",m.get("spName"));

for (Map seal : sealCtInfo){

if(seal.get("group").equals("get")){

if(m.get("spNo").equals(seal.get("spNo"))) {

Long countNum = (Long) seal.get("countNum");

mr.put("sealGetCount", countNum!=0l?countNum:0);

}

}else if(seal.get("group").equals("completed")){

if(m.get("spNo").equals(seal.get("spNo"))) {

Long countNum = (Long) seal.get("countNum");

mr.put("sealSuccessCount", countNum!=0l?countNum:0);

}

}

}

for (Map stamp : stampCtInfo){

if(stamp.get("group").equals("get")) {

if(m.get("spNo").equals(stamp.get("spNo"))) {

Long countNum = (Long) stamp.get("countNum");

mr.put("stampGetCount", countNum!=0l?countNum:0);

}

}else if(stamp.get("group").equals("completed")){

if(m.get("spNo").equals(stamp.get("spNo"))) {

Long countNum = (Long) stamp.get("countNum");

mr.put("stampSuccessCount", countNum!=0l?countNum:0);

}

}

}

orderCountVoList.add(mr);

}

}

logger.info("orderCountVoList"+orderCountVoList);

pageBean.setData(orderCountVoList);

pageBean.setCount(page.getTotal());

return pageBean;

}

public PageBean findAppOrderList(PageBean pageBean, SzgSealOrderVo szgSealOrderVo){

if(pageBean==null)

pageBean = new PageBean();

Page page = PageHelper.startPage(pageBean.getPage(),pageBean.getPageSize());

List<Map> appList = cbAppMapper.getAPPList(szgSealOrderVo.getAppId());

List<Map> sealCtInfo = szgSealOrderMapper.getSealAppInfo(szgSealOrderVo);

SzgStampOrderVo szgStampOrderVo = new SzgStampOrderVo();

szgStampOrderVo.setStartTime(szgSealOrderVo.getStartTime());

szgStampOrderVo.setEndTime(szgSealOrderVo.getEndTime());

List<Map> stampCtInfo = szgStampOrderMapper.getStampAppInfo(szgStampOrderVo);

List orderCountVoList = new ArrayList();

if((sealCtInfo!=null && !sealCtInfo.isEmpty()) || (stampCtInfo != null && !stampCtInfo.isEmpty())){

for (Map m : appList){

Map mr = new HashMap();

mr.put("appId",m.get("appId"));

mr.put("appName",m.get("appName"));

for (Map seal : sealCtInfo){

if(seal.get("group").equals("get")){

if(m.get("appId").equals(seal.get("appId"))) {

Long countNum = (Long) seal.get("countNum");

mr.put("sealGetCount", countNum!=0l?countNum:0);

}

}else if(seal.get("group").equals("completed")){

if(m.get("appId").equals(seal.get("appId"))) {

Long countNum = (Long) seal.get("countNum");

mr.put("sealSuccessCount", countNum!=0l?countNum:0);

}

}

}

NumberFormat numberFormat = NumberFormat.getInstance();

// 设置精确到小数点后2位

numberFormat.setMaximumFractionDigits(2);

String sealTotal = "0";

String sealSuc = "0";

String sealResult = "0";

if(mr.get("sealSuccessCount") != null && mr.get("sealSuccessCount") != ""){

sealSuc = mr.get("sealSuccessCount").toString();

}

if(mr.get("sealGetCount") != null && mr.get("sealGetCount") != ""){

sealTotal = mr.get("sealGetCount").toString();

}

if(sealTotal!="0" && sealTotal != null){

sealResult = numberFormat.format(Float.parseFloat(sealSuc) / Float.parseFloat(sealTotal) \* 100);

mr.put("sealSucPercentage",sealResult+"%");

}else{

mr.put("sealSucPercentage","0.00%");

}

for (Map stamp : stampCtInfo){

if(stamp.get("group").equals("get")){

if(m.get("appId").equals(stamp.get("appId"))) {

Long countNum = (Long) stamp.get("countNum");

mr.put("stampGetCount", countNum!=0l?countNum:0);

}

}else if(stamp.get("group").equals("completed")){

if(m.get("appId").equals(stamp.get("appId"))) {

Long countNum = (Long) stamp.get("countNum");

mr.put("stampSuccessCount", countNum!=0l?countNum:0);

}

}

}

String stampTotal = "0";

String stampSuc = "0";

String stampResult = "0";

if(mr.get("stampSuccessCount") != null && mr.get("stampSuccessCount") != ""){

stampSuc = mr.get("stampSuccessCount").toString();

}

if(mr.get("stampGetCount") != null && mr.get("stampGetCount") != ""){

stampTotal = mr.get("stampGetCount").toString() ;

}

if(stampTotal!="0" && stampTotal != null){

stampResult = numberFormat.format(Float.parseFloat(stampSuc) / Float.parseFloat(stampTotal) \* 100);

mr.put("stampSucPercentage",stampResult+"%");

}else{

mr.put("stampSucPercentage","0.00%");

}

orderCountVoList.add(mr);

}

}

logger.info("orderCountVoList"+orderCountVoList);

pageBean.setData(orderCountVoList);

pageBean.setCount(page.getTotal());

return pageBean;

}

}

package cn.g4b.fm.service.szg;

import cn.g4b.fm.common.model.PageBean;

import cn.g4b.fm.common.util.StringUtils;

import cn.g4b.fm.dao.szg.SzgApiMapper;

import cn.g4b.fm.dao.szg.SzgAppApiMapper;

import cn.g4b.fm.model.sys.CbApp;

import cn.g4b.fm.model.szg.SzgApi;

import cn.g4b.fm.model.szg.SzgApiWithBLOBs;

import com.github.pagehelper.Page;

import com.github.pagehelper.PageHelper;

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

import org.springframework.stereotype.Service;

import org.springframework.transaction.annotation.Transactional;

import java.util.List;

@Service

public class SzgApiService {

@Autowired

private SzgApiMapper apiMapper;

public PageBean<SzgApi> findPageList(PageBean<SzgApi> pageBean, SzgApi api) {

if(pageBean==null)

pageBean = new PageBean();

Page page = PageHelper.startPage(pageBean.getPage(),pageBean.getPageSize());

List<SzgApi> list = apiMapper.findList(api);

pageBean.setData(list);

pageBean.setCount(page.getTotal());

return pageBean;

}

public SzgApiWithBLOBs getApiInfoWithBlobs(Integer id) {

if(id == null)

return null;

return apiMapper.selectByPrimaryKey(id);

}

@Transactional(rollbackFor = Exception.class)

public void saveApi(SzgApiWithBLOBs api) {

if(api.getId() == null){

apiMapper.insert(api);

}else{

apiMapper.updateByPrimaryKeyWithBLOBs(api);

}

}

public PageBean<SzgApi> findListForUseObj(PageBean<SzgApi> pageBean, String useObj) {

if(pageBean==null)

pageBean = new PageBean();

Page page = PageHelper.startPage(pageBean.getPage(),pageBean.getPageSize());

if(!StringUtils.isEmpty(useObj)){

useObj = useObj.replaceAll("\_",",");

}

List<SzgApi> list = apiMapper.findByUseObj(useObj);

pageBean.setData(list);

pageBean.setCount(page.getTotal());

return pageBean;

}

}

package cn.g4b.fm.service.szg;

import cn.g4b.fm.common.model.BizException;

import cn.g4b.fm.common.model.CommonEnum;

import cn.g4b.fm.common.model.ResultBody;

import cn.g4b.fm.common.util.JacksonUtil;

import cn.g4b.fm.common.util.ParamUtil;

import cn.g4b.fm.common.util.StringUtils;

import cn.g4b.fm.dao.szg.SzgHksecurityOrderMapper;

import cn.g4b.fm.model.sys.CbApp;

import cn.g4b.fm.model.szg.SzgHksecurityOrder;

import cn.g4b.fm.model.szg.api.CertOrderRequest;

import cn.g4b.fm.service.tbaas.HksecurityChaincodeApiService;

import com.fasterxml.jackson.databind.DeserializationFeature;

import com.fasterxml.jackson.databind.JsonNode;

import com.fasterxml.jackson.databind.ObjectMapper;

import com.tencent.cloud.tbaas.bean.FabricQueryResponse;

import com.tencent.cloud.tbaas.bean.FabricTransactResponse;

import org.hyperledger.fabric.sdk.exception.InvalidArgumentException;

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

import org.springframework.mail.MailParseException;

import org.springframework.stereotype.Service;

import org.springframework.transaction.annotation.Transactional;

import java.io.IOException;

import java.util.Date;

import java.util.Map;

@Service

public class SzgHksecurityOrderService {

@Autowired

private SzgHksecurityOrderMapper hksecurityOrderMapper;

@Autowired

private HksecurityChaincodeApiService hksecurityChaincodeApiService;

@Transactional(rollbackFor = Exception.class)

public SzgHksecurityOrder addHkSecurityContractData(String requestBody, CbApp appInfo) {

if(StringUtils.isEmpty(requestBody))

throw new BizException(CommonEnum.ERROR\_NULL\_PARAMS,"订单参数不能为null");

ObjectMapper objectMapper = new ObjectMapper();

objectMapper.configure(DeserializationFeature.FAIL\_ON\_UNKNOWN\_PROPERTIES, false);

try {

SzgHksecurityOrder hksecurityOrder = objectMapper.readValue(requestBody, SzgHksecurityOrder.class);

//检查参数是否完整

ParamUtil.nullCheck(hksecurityOrder,"orderNo","contractHash");

FabricTransactResponse response = hksecurityChaincodeApiService.addHkSecurityContractData(hksecurityOrder);

if(response == null){

throw new BizException(CommonEnum.INTERNAL\_SERVER\_ERROR,"调用tbaas失败");

}

if(!"SUCCESS".equals(response.getCode())){

throw new BizException(CommonEnum.INTERNAL\_SERVER\_ERROR,"调用tbaas失败:"+response.getMessage());

}

String data = response.getData().toString();

JsonNode jsonNode = objectMapper.readTree(data);

if(!jsonNode.get("Result").asBoolean()){

throw new BizException(CommonEnum.INTERNAL\_SERVER\_ERROR,"调用tbaas失败:"+jsonNode.get("Msg").toString());

}

String txId = response.getTxId();

hksecurityOrder.setTxId(txId);

hksecurityOrder.setCreateTime(new Date());

hksecurityOrder.setAppId(appInfo.getAppId());

hksecurityOrderMapper.insert(hksecurityOrder);

return hksecurityOrder;

} catch (IOException e) {

e.printStackTrace();

throw new BizException(CommonEnum.PARAMS\_CHECK\_ERR,e.getMessage());

} catch (IllegalAccessException |NoSuchFieldException | InvalidArgumentException e) {

e.printStackTrace();

throw new BizException(CommonEnum.PARAMS\_CHECK\_ERR,e.getMessage());

}

}

public Object queryHkSecurityContractData(String requestBody) {

if(StringUtils.isEmpty(requestBody))

throw new BizException(CommonEnum.ERROR\_NULL\_PARAMS,"参数不能为null");

ObjectMapper objectMapper = new ObjectMapper();

try {

JsonNode jsonNode = objectMapper.readTree(requestBody);

String orderNo = jsonNode.get("orderNo").asText();

FabricQueryResponse response = hksecurityChaincodeApiService.queryHkSecurityContractData(orderNo);

if(!"SUCCESS".equals(response.getCode()))

throw new BizException(CommonEnum.INTERNAL\_SERVER\_ERROR,"调用tbaas失败："+response.getMessage());

String data = response.getData().toString();

Object orderInfo = JacksonUtil.parseTbaasResponseBody(data);

if(orderInfo instanceof Map){

Map m = (Map)orderInfo;

return m.get("order");

}

return orderInfo;

} catch (IOException | InvalidArgumentException e) {

e.printStackTrace();

throw new BizException(CommonEnum.BODY\_NOT\_MATCH,e.getMessage());

}

}

}

package cn.g4b.fm.service.szg;

import cn.g4b.fm.common.model.BizException;

import cn.g4b.fm.common.model.CommonEnum;

import cn.g4b.fm.common.util.MD5Utils;

import cn.g4b.fm.common.util.ParamUtil;

import cn.g4b.fm.common.util.StringUtils;

import cn.g4b.fm.dao.szg.SzgSignaturePicMapper;

import cn.g4b.fm.model.sys.CbApp;

import cn.g4b.fm.model.szg.SzgSignaturePic;

import cn.g4b.fm.service.tbaas.LogoChaincodeApiService;

import com.fasterxml.jackson.databind.DeserializationFeature;

import com.fasterxml.jackson.databind.JsonNode;

import com.fasterxml.jackson.databind.ObjectMapper;

import com.tencent.cloud.tbaas.bean.FabricTransactResponse;

import org.bouncycastle.util.encoders.Base64;

import org.hyperledger.fabric.sdk.exception.InvalidArgumentException;

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

import org.springframework.stereotype.Service;

import java.io.IOException;

import java.util.Date;

import java.util.List;

import java.util.Map;

@Service

public class SzgSignaturePicService {

@Autowired

private SzgSignaturePicMapper signaturePicMapper;

@Autowired

private FastDFSService fastDFSService;

@Autowired

private LogoChaincodeApiService chaincodeApiService;

public SzgSignaturePic addSignaturePic(String requestBody, CbApp appInfo) {

if(StringUtils.isEmpty(requestBody))

throw new BizException(CommonEnum.ERROR\_NULL\_PARAMS,"参数不能为null");

ObjectMapper objectMapper = new ObjectMapper();

try {

objectMapper.configure(DeserializationFeature.FAIL\_ON\_UNKNOWN\_PROPERTIES, false);

SzgSignaturePic signaturePic = objectMapper.readValue(requestBody, SzgSignaturePic.class);

ParamUtil.nullCheck(signaturePic,"userName","identityType","identityNo","sealType","signaturePicFile");

byte state = 0;

signaturePic.setState(state);

SzgSignaturePic tmp = signaturePicMapper.getSignaturePic(signaturePic);

if(tmp != null)

throw new BizException(CommonEnum.PARAMS\_CHECK\_ERR,"不能提交重复的手写签名");

//印模上传文件服务器

byte[] fileBytes = Base64.decode(signaturePic.getSignaturePicFile());

String signaturePicFileUrl = fastDFSService.uploadFile("logo", fileBytes, signaturePic.getIdentityNo() + ".png");

String signaturePicFileHash = MD5Utils.MD5Encode(fileBytes);

signaturePic.setSignaturePicFileUrl(signaturePicFileUrl);

signaturePic.setSignaturePicFileHash(signaturePicFileHash);

signaturePic.setCreateTime(new Date());

signaturePic.setAppId(appInfo.getAppId());

FabricTransactResponse response = chaincodeApiService.addSignaturePic(signaturePic);

if(!"SUCCESS".equals(response.getCode())){

throw new BizException(CommonEnum.INTERNAL\_SERVER\_ERROR,response.getMessage());

}

signaturePicMapper.insert(signaturePic);

return signaturePic;

} catch (IOException e) {

e.printStackTrace();

throw new BizException(CommonEnum.INTERNAL\_SERVER\_ERROR,e.getMessage());

} catch (IllegalAccessException e) {

e.printStackTrace();

throw new BizException(CommonEnum.INTERNAL\_SERVER\_ERROR,e.getMessage());

} catch (NoSuchFieldException e) {

e.printStackTrace();

throw new BizException(CommonEnum.INTERNAL\_SERVER\_ERROR,e.getMessage());

} catch (InvalidArgumentException e) {

e.printStackTrace();

throw new BizException(CommonEnum.INTERNAL\_SERVER\_ERROR,e.getMessage());

}

}

public List<Map> querySignatureList(String requestBody, CbApp appInfo) throws IOException {

if(StringUtils.isEmpty(requestBody))

throw new BizException(CommonEnum.ERROR\_NULL\_PARAMS,"参数不能为null");

ObjectMapper objectMapper = new ObjectMapper();

JsonNode jsonNode = objectMapper.readTree(requestBody);

Integer identityType = jsonNode.get("identityType").asInt();

String identityNo = jsonNode.get("identityNo").asText();

if(identityType == null){

throw new BizException(CommonEnum.PARAMS\_CHECK\_ERR,"identityType不能为null");

}

if(identityNo == null){

throw new BizException(CommonEnum.PARAMS\_CHECK\_ERR,"identityNo不能为null");

}

return signaturePicMapper.querySignatureList(identityType,identityNo);

}

public void disableSignaturePic(String requestBody, CbApp appInfo) throws IOException, InvalidArgumentException {

if(StringUtils.isEmpty(requestBody))

throw new BizException(CommonEnum.ERROR\_NULL\_PARAMS,"参数不能为null");

ObjectMapper objectMapper = new ObjectMapper();

JsonNode jsonNode = objectMapper.readTree(requestBody);

Integer identityType = jsonNode.get("identityType").asInt();

String identityNo = jsonNode.get("identityNo").asText();

Integer sealType = jsonNode.get("sealType").asInt();

if(identityType == null){

throw new BizException(CommonEnum.PARAMS\_CHECK\_ERR,"identityType不能为null");

}

if(identityNo == null){

throw new BizException(CommonEnum.PARAMS\_CHECK\_ERR,"identityNo不能为null");

}

if(sealType == null){

throw new BizException(CommonEnum.PARAMS\_CHECK\_ERR,"sealType不能为null");

}

FabricTransactResponse response = chaincodeApiService.disableSignaturePic(identityType,identityNo,sealType);

if(response == null){

throw new BizException(CommonEnum.INTERNAL\_SERVER\_ERROR,"调用tbaas失败");

}

if(!"SUCCESS".equals(response.getCode()) && response.getMessage().toString().indexOf("该手写签名图片已是失效状态")==-1){

throw new BizException(CommonEnum.INTERNAL\_SERVER\_ERROR,"调用tbaas失败:"+response.getMessage());

}

signaturePicMapper.disableSignature(identityType,identityNo,sealType);

}

}

<?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="cn.g4b.fm.dao.szg.CbAppExtendInfoMapper">

<resultMap id="BaseResultMap" type="cn.g4b.fm.model.szg.CbAppExtendInfo">

<result column="app\_id" jdbcType="VARCHAR" property="appId" />

<result column="dep\_name" jdbcType="VARCHAR" property="depName" />

<result column="ent\_code" jdbcType="VARCHAR" property="entCode" />

<result column="link\_address" jdbcType="VARCHAR" property="linkAddress" />

<result column="link\_person" jdbcType="VARCHAR" property="linkPerson" />

<result column="dep\_post" jdbcType="VARCHAR" property="depPost" />

<result column="link\_phone" jdbcType="VARCHAR" property="linkPhone" />

<result column="email" jdbcType="VARCHAR" property="email" />

<result column="ent\_nature" jdbcType="INTEGER" property="entNature" />

<result column="net\_nature" jdbcType="INTEGER" property="netNature" />

<result column="ip" jdbcType="VARCHAR" property="ip" />

<result column="app\_demand" jdbcType="VARCHAR" property="appDemand" />

<result column="user\_number" jdbcType="VARCHAR" property="userNumber" />

<result column="concurrency" jdbcType="VARCHAR" property="concurrency" />

<result column="remark" jdbcType="VARCHAR" property="remark" />

<result column="audit\_time" jdbcType="TIMESTAMP" property="auditTime" />

<result column="user\_id" jdbcType="VARCHAR" property="userId" />

<result column="hash\_value" jdbcType="VARCHAR" property="hashValue" />

<result column="app\_name" jdbcType="VARCHAR" property="appName" />

<result column="app\_desc" jdbcType="VARCHAR" property="appDesc" />

<result column="app\_code" jdbcType="VARCHAR" property="appCode" />

<result column="app\_type" jdbcType="TINYINT" property="appType" />

<result column="callback\_url" jdbcType="VARCHAR" property="callbackUrl" />

<result column="status" jdbcType="INTEGER" property="status" />

<result column="is\_deliver" jdbcType="INTEGER" property="isDeliver" />

</resultMap>

<select id="getExtendInfoByAppId" resultMap="BaseResultMap" parameterType="java.lang.String">

select \* from cb\_app\_extend\_info where app\_id = #{appId,jdbcType=VARCHAR}

</select>

<insert id="insert" parameterType="cn.g4b.fm.model.szg.CbAppExtendInfo">

insert into cb\_app\_extend\_info (app\_id, dep\_name, ent\_code,

link\_address, link\_person, dep\_post,

link\_phone, email, ent\_nature,

net\_nature, ip, app\_demand,

user\_number, concurrency, remark,audit\_time

)

values (#{appId,jdbcType=VARCHAR}, #{depName,jdbcType=VARCHAR}, #{entCode,jdbcType=VARCHAR},

#{linkAddress,jdbcType=VARCHAR}, #{linkPerson,jdbcType=VARCHAR}, #{depPost,jdbcType=VARCHAR},

#{linkPhone,jdbcType=VARCHAR}, #{email,jdbcType=VARCHAR}, #{entNature,jdbcType=INTEGER},

#{netNature,jdbcType=INTEGER}, #{ip,jdbcType=VARCHAR}, #{appDemand,jdbcType=VARCHAR},

#{userNumber,jdbcType=VARCHAR}, #{concurrency,jdbcType=VARCHAR}, #{remark,jdbcType=VARCHAR},#{auditTime,jdbcType=TIMESTAMP}

)

</insert>

<insert id="insertSelective" parameterType="cn.g4b.fm.model.szg.CbAppExtendInfo">

insert into cb\_app\_extend\_info

<trim prefix="(" suffix=")" suffixOverrides=",">

<if test="appId != null">

app\_id,

</if>

<if test="depName != null">

dep\_name,

</if>

<if test="entCode != null">

ent\_code,

</if>

<if test="linkAddress != null">

link\_address,

</if>

<if test="linkPerson != null">

link\_person,

</if>

<if test="depPost != null">

dep\_post,

</if>

<if test="linkPhone != null">

link\_phone,

</if>

<if test="email != null">

email,

</if>

<if test="entNature != null">

ent\_nature,

</if>

<if test="netNature != null">

net\_nature,

</if>

<if test="ip != null">

ip,

</if>

<if test="appDemand != null">

app\_demand,

</if>

<if test="userNumber != null">

user\_number,

</if>

<if test="concurrency != null">

concurrency,

</if>

<if test="remark != null">

remark,

</if>

</trim>

<trim prefix="values (" suffix=")" suffixOverrides=",">

<if test="appId != null">

#{appId,jdbcType=VARCHAR},

</if>

<if test="depName != null">

#{depName,jdbcType=VARCHAR},

</if>

<if test="entCode != null">

#{entCode,jdbcType=VARCHAR},

</if>

<if test="linkAddress != null">

#{linkAddress,jdbcType=VARCHAR},

</if>

<if test="linkPerson != null">

#{linkPerson,jdbcType=VARCHAR},

</if>

<if test="depPost != null">

#{depPost,jdbcType=VARCHAR},

</if>

<if test="linkPhone != null">

#{linkPhone,jdbcType=VARCHAR},

</if>

<if test="email != null">

#{email,jdbcType=VARCHAR},

</if>

<if test="entNature != null">

#{entNature,jdbcType=INTEGER},

</if>

<if test="netNature != null">

#{netNature,jdbcType=INTEGER},

</if>

<if test="ip != null">

#{ip,jdbcType=VARCHAR},

</if>

<if test="appDemand != null">

#{appDemand,jdbcType=VARCHAR},

</if>

<if test="userNumber != null">

#{userNumber,jdbcType=VARCHAR},

</if>

<if test="concurrency != null">

#{concurrency,jdbcType=VARCHAR},

</if>

<if test="remark != null">

#{remark,jdbcType=VARCHAR},

</if>

</trim>

</insert>

<select id="findList" resultMap="BaseResultMap" parameterType="cn.g4b.fm.model.szg.CbAppExtendInfo">

SELECT \* FROM cb\_app LEFT JOIN cb\_app\_extend\_info on (cb\_app.app\_id = cb\_app\_extend\_info.app\_id)

</select>

</mapper>

<?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="cn.g4b.fm.dao.szg.SzgApiMapper">

<resultMap id="BaseResultMap" type="cn.g4b.fm.model.szg.SzgApi">

<id column="id" jdbcType="INTEGER" property="id" />

<result column="api\_name" jdbcType="VARCHAR" property="apiName" />

<result column="api\_desc" jdbcType="VARCHAR" property="apiDesc" />

<result column="api\_url" jdbcType="VARCHAR" property="apiUrl" />

<result column="request\_type" jdbcType="VARCHAR" property="requestType" />

<result column="status" jdbcType="INTEGER" property="status" />

<result column="use\_obj" jdbcType="TINYINT" property="useObj" />

</resultMap>

<resultMap extends="BaseResultMap" id="ResultMapWithBLOBs" type="cn.g4b.fm.model.szg.SzgApiWithBLOBs">

<result column="request\_header" jdbcType="LONGVARCHAR" property="requestHeader" />

<result column="request\_body" jdbcType="LONGVARCHAR" property="requestBody" />

<result column="response\_body" jdbcType="LONGVARCHAR" property="responseBody" />

</resultMap>

<sql id="Base\_Column\_List">

id, api\_name, api\_desc, api\_url, request\_type, `status`, use\_obj

</sql>

<sql id="Blob\_Column\_List">

request\_header, request\_body, response\_body

</sql>

<select id="selectByPrimaryKey" parameterType="java.lang.Integer" resultMap="ResultMapWithBLOBs">

select

<include refid="Base\_Column\_List" />

,

<include refid="Blob\_Column\_List" />

from szg\_api

where id = #{id,jdbcType=INTEGER}

</select>

<select id="findList" parameterType="cn.g4b.fm.model.szg.SzgApi" resultMap="BaseResultMap">

select

<include refid="Base\_Column\_List" />

from szg\_api

where 1=1

<if test="status != null">

and status = #{status,jdbcType=INTEGER}

</if>

<if test="useObj != null">

and use\_obj = #{useObj,jdbcType=TINYINT}

</if>

</select>

<select id="findByUseObj" parameterType="java.lang.String" resultMap="BaseResultMap">

select

<include refid="Base\_Column\_List" />

from szg\_api

where

status = 0

<if test="useObj != null">

and use\_obj in (${useObj})

</if>

</select>

<select id="getApiListByAppId" resultType="java.util.Map">

SELECT id,api\_name as apiName,api\_url as apiUrl,app\_id as appId from szg\_api

LEFT JOIN szg\_app\_api ON szg\_api.id = szg\_app\_api.api\_id

and app\_id=#{appId,jdbcType=VARCHAR} where status = 0

</select>

<select id="getAppApiList" resultType="java.lang.String">

select api\_url from szg\_api where EXISTS (select 1 from szg\_app\_api where api\_id = id and app\_id=#{appId,jdbcType=VARCHAR})

</select>

<delete id="deleteByPrimaryKey" parameterType="java.lang.Integer">

delete from szg\_api

where id = #{id,jdbcType=INTEGER}

</delete>

<insert id="insert" parameterType="cn.g4b.fm.model.szg.SzgApiWithBLOBs">

insert into szg\_api (id, api\_name, api\_desc,

api\_url, request\_type, `status`,

use\_obj, request\_header, request\_body,

response\_body)

values (#{id,jdbcType=INTEGER}, #{apiName,jdbcType=VARCHAR}, #{apiDesc,jdbcType=VARCHAR},

#{apiUrl,jdbcType=VARCHAR}, #{requestType,jdbcType=VARCHAR}, #{status,jdbcType=INTEGER},

#{useObj,jdbcType=TINYINT}, #{requestHeader,jdbcType=LONGVARCHAR}, #{requestBody,jdbcType=LONGVARCHAR},

#{responseBody,jdbcType=LONGVARCHAR})

</insert>

<insert id="insertSelective" parameterType="cn.g4b.fm.model.szg.SzgApiWithBLOBs">

insert into szg\_api

<trim prefix="(" suffix=")" suffixOverrides=",">

<if test="id != null">

id,

</if>

<if test="apiName != null">

api\_name,

</if>

<if test="apiDesc != null">

api\_desc,

</if>

<if test="apiUrl != null">

api\_url,

</if>

<if test="requestType != null">

request\_type,

</if>

<if test="status != null">

`status`,

</if>

<if test="useObj != null">

use\_obj,

</if>

<if test="requestHeader != null">

request\_header,

</if>

<if test="requestBody != null">

request\_body,

</if>

<if test="responseBody != null">

response\_body,

</if>

</trim>

<trim prefix="values (" suffix=")" suffixOverrides=",">

<if test="id != null">

#{id,jdbcType=INTEGER},

</if>

<if test="apiName != null">

#{apiName,jdbcType=VARCHAR},

</if>

<if test="apiDesc != null">

#{apiDesc,jdbcType=VARCHAR},

</if>

<if test="apiUrl != null">

#{apiUrl,jdbcType=VARCHAR},

</if>

<if test="requestType != null">

#{requestType,jdbcType=VARCHAR},

</if>

<if test="status != null">

#{status,jdbcType=INTEGER},

</if>

<if test="useObj != null">

#{useObj,jdbcType=TINYINT},

</if>

<if test="requestHeader != null">

#{requestHeader,jdbcType=LONGVARCHAR},

</if>

<if test="requestBody != null">

#{requestBody,jdbcType=LONGVARCHAR},

</if>

<if test="responseBody != null">

#{responseBody,jdbcType=LONGVARCHAR},

</if>

</trim>

</insert>

<update id="updateByPrimaryKeySelective" parameterType="cn.g4b.fm.model.szg.SzgApiWithBLOBs">

update szg\_api

<set>

<if test="apiName != null">

api\_name = #{apiName,jdbcType=VARCHAR},

</if>

<if test="apiDesc != null">

api\_desc = #{apiDesc,jdbcType=VARCHAR},

</if>

<if test="apiUrl != null">

api\_url = #{apiUrl,jdbcType=VARCHAR},

</if>

<if test="requestType != null">

request\_type = #{requestType,jdbcType=VARCHAR},

</if>

<if test="status != null">

`status` = #{status,jdbcType=INTEGER},

</if>

<if test="useObj != null">

use\_obj = #{useObj,jdbcType=TINYINT},

</if>

<if test="requestHeader != null">

request\_header = #{requestHeader,jdbcType=LONGVARCHAR},

</if>

<if test="requestBody != null">

request\_body = #{requestBody,jdbcType=LONGVARCHAR},

</if>

<if test="responseBody != null">

response\_body = #{responseBody,jdbcType=LONGVARCHAR},

</if>

</set>

where id = #{id,jdbcType=INTEGER}

</update>

<update id="updateByPrimaryKeyWithBLOBs" parameterType="cn.g4b.fm.model.szg.SzgApiWithBLOBs">

update szg\_api

set api\_name = #{apiName,jdbcType=VARCHAR},

api\_desc = #{apiDesc,jdbcType=VARCHAR},

api\_url = #{apiUrl,jdbcType=VARCHAR},

request\_type = #{requestType,jdbcType=VARCHAR},

`status` = #{status,jdbcType=INTEGER},

use\_obj = #{useObj,jdbcType=TINYINT},

request\_header = #{requestHeader,jdbcType=LONGVARCHAR},

request\_body = #{requestBody,jdbcType=LONGVARCHAR},

response\_body = #{responseBody,jdbcType=LONGVARCHAR}

where id = #{id,jdbcType=INTEGER}

</update>

<update id="updateByPrimaryKey" parameterType="cn.g4b.fm.model.szg.SzgApi">

update szg\_api

set api\_name = #{apiName,jdbcType=VARCHAR},

api\_desc = #{apiDesc,jdbcType=VARCHAR},

api\_url = #{apiUrl,jdbcType=VARCHAR},

request\_type = #{requestType,jdbcType=VARCHAR},

`status` = #{status,jdbcType=INTEGER},

use\_obj = #{useObj,jdbcType=TINYINT}

where id = #{id,jdbcType=INTEGER}

</update>

</mapper>

<?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="cn.g4b.fm.dao.szg.SzgEventMapper">

<resultMap id="BaseResultMap" type="cn.g4b.fm.model.szg.SzgEvent">

<id column="id" jdbcType="INTEGER" property="id" />

<result column="event\_name" jdbcType="VARCHAR" property="eventName" />

<result column="event\_index" jdbcType="INTEGER" property="eventIndex" />

<result column="update\_time" jdbcType="TIMESTAMP" property="updateTime" />

</resultMap>

<sql id="Base\_Column\_List">

id, event\_name, event\_index, update\_time

</sql>

<select id="selectByPrimaryKey" parameterType="java.lang.Integer" resultMap="BaseResultMap">

select

<include refid="Base\_Column\_List" />

from szg\_event

where id = #{id,jdbcType=INTEGER}

</select>

<select id="selectByEventName" parameterType="java.lang.String" resultMap="BaseResultMap">

select

<include refid="Base\_Column\_List" />

from szg\_event

where event\_name = #{eventName,jdbcType=VARCHAR}

</select>

<delete id="deleteByPrimaryKey" parameterType="java.lang.Integer">

delete from szg\_event

where id = #{id,jdbcType=INTEGER}

</delete>

<insert id="insert" parameterType="cn.g4b.fm.model.szg.SzgEvent">

insert into szg\_event (id, event\_name, event\_index,

update\_time)

values (#{id,jdbcType=INTEGER}, #{eventName,jdbcType=VARCHAR}, #{eventIndex,jdbcType=INTEGER},

#{updateTime,jdbcType=TIMESTAMP})

</insert>

<insert id="insertSelective" parameterType="cn.g4b.fm.model.szg.SzgEvent">

insert into szg\_event

<trim prefix="(" suffix=")" suffixOverrides=",">

<if test="id != null">

id,

</if>

<if test="eventName != null">

event\_name,

</if>

<if test="eventIndex != null">

event\_index,

</if>

<if test="updateTime != null">

update\_time,

</if>

</trim>

<trim prefix="values (" suffix=")" suffixOverrides=",">

<if test="id != null">

#{id,jdbcType=INTEGER},

</if>

<if test="eventName != null">

#{eventName,jdbcType=VARCHAR},

</if>

<if test="eventIndex != null">

#{eventIndex,jdbcType=INTEGER},

</if>

<if test="updateTime != null">

#{updateTime,jdbcType=TIMESTAMP},

</if>

</trim>

</insert>

<update id="updateByPrimaryKeySelective" parameterType="cn.g4b.fm.model.szg.SzgEvent">

update szg\_event

<set>

<if test="eventName != null">

event\_name = #{eventName,jdbcType=VARCHAR},

</if>

<if test="eventIndex != null">

event\_index = #{eventIndex,jdbcType=INTEGER},

</if>

<if test="updateTime != null">

update\_time = #{updateTime,jdbcType=TIMESTAMP},

</if>

</set>

where id = #{id,jdbcType=INTEGER}

</update>

<update id="updateByPrimaryKey" parameterType="cn.g4b.fm.model.szg.SzgEvent">

update szg\_event

set event\_name = #{eventName,jdbcType=VARCHAR},

event\_index = #{eventIndex,jdbcType=INTEGER},

update\_time = #{updateTime,jdbcType=TIMESTAMP}

where id = #{id,jdbcType=INTEGER}

</update>

<update id="updateByIndex" >

update szg\_event

set

event\_index = #{newIndex,jdbcType=INTEGER},

update\_time = NOW()

where id = #{id,jdbcType=INTEGER}

</update>

</mapper>

<?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="cn.g4b.fm.dao.szg.SzgGzStampingMapper">

<resultMap id="BaseResultMap" type="cn.g4b.fm.model.szg.SzgGzStamping">

<id column="tx\_id" jdbcType="VARCHAR" property="txId" />

<result column="record\_id" jdbcType="VARCHAR" property="recordId" />

<result column="agent\_name" jdbcType="VARCHAR" property="agentName" />

<result column="agent\_identity\_type" jdbcType="TINYINT" property="agentIdentityType" />

<result column="agent\_identity\_no" jdbcType="VARCHAR" property="agentIdentityNo" />

<result column="agent\_photo\_url" jdbcType="VARCHAR" property="agentPhotoUrl" />

<result column="agent\_photo\_hash" jdbcType="VARCHAR" property="agentPhotoHash" />

<result column="agent\_verify\_sp\_no" jdbcType="VARCHAR" property="agentVerifySpNo" />

<result column="agent\_verify\_order\_no" jdbcType="VARCHAR" property="agentVerifyOrderNo" />

<result column="seal\_code" jdbcType="VARCHAR" property="sealCode" />

<result column="seal\_user\_cert\_no" jdbcType="VARCHAR" property="sealUserCertNo" />

<result column="original\_file\_hash" jdbcType="VARCHAR" property="originalFileHash" />

<result column="stamping\_sign" jdbcType="VARCHAR" property="stampingSign" />

<result column="stamping\_result\_file\_hash" jdbcType="VARCHAR" property="stampingResultFileHash" />

<result column="stamping\_result" jdbcType="TINYINT" property="stampingResult" />

<result column="stamping\_msg" jdbcType="VARCHAR" property="stampingMsg" />

<result column="stamping\_time" jdbcType="BIGINT" property="stampingTime" />

<result column="app\_id" jdbcType="VARCHAR" property="appId" />

</resultMap>

<sql id="Base\_Column\_List">

tx\_id, record\_id, agent\_name, agent\_identity\_type, agent\_identity\_no, agent\_photo\_url,

agent\_photo\_hash, agent\_verify\_sp\_no, agent\_verify\_order\_no, seal\_code, seal\_user\_cert\_no,

original\_file\_hash, stamping\_sign, stamping\_result\_file\_hash, stamping\_result, stamping\_msg,

stamping\_time, app\_id

</sql>

<select id="selectByPrimaryKey" parameterType="java.lang.String" resultMap="BaseResultMap">

select

<include refid="Base\_Column\_List" />

from szg\_gz\_stamping

where tx\_id = #{txId,jdbcType=VARCHAR}

</select>

<delete id="deleteByPrimaryKey" parameterType="java.lang.String">

delete from szg\_gz\_stamping

where tx\_id = #{txId,jdbcType=VARCHAR}

</delete>

<insert id="insert" parameterType="cn.g4b.fm.model.szg.SzgGzStamping">

insert into szg\_gz\_stamping (tx\_id, record\_id, agent\_name,

agent\_identity\_type, agent\_identity\_no, agent\_photo\_url,

agent\_photo\_hash, agent\_verify\_sp\_no, agent\_verify\_order\_no,

seal\_code, seal\_user\_cert\_no, original\_file\_hash,

stamping\_sign, stamping\_result\_file\_hash, stamping\_result,

stamping\_msg, stamping\_time, app\_id

)

values (#{txId,jdbcType=VARCHAR}, #{recordId,jdbcType=VARCHAR}, #{agentName,jdbcType=VARCHAR},

#{agentIdentityType,jdbcType=TINYINT}, #{agentIdentityNo,jdbcType=VARCHAR}, #{agentPhotoUrl,jdbcType=VARCHAR},

#{agentPhotoHash,jdbcType=VARCHAR}, #{agentVerifySpNo,jdbcType=VARCHAR}, #{agentVerifyOrderNo,jdbcType=VARCHAR},

#{sealCode,jdbcType=VARCHAR}, #{sealUserCertNo,jdbcType=VARCHAR}, #{originalFileHash,jdbcType=VARCHAR},

#{stampingSign,jdbcType=VARCHAR}, #{stampingResultFileHash,jdbcType=VARCHAR}, #{stampingResult,jdbcType=TINYINT},

#{stampingMsg,jdbcType=VARCHAR}, #{stampingTime,jdbcType=BIGINT}, #{appId,jdbcType=VARCHAR}

)

</insert>

<insert id="insertSelective" parameterType="cn.g4b.fm.model.szg.SzgGzStamping">

insert into szg\_gz\_stamping

<trim prefix="(" suffix=")" suffixOverrides=",">

<if test="txId != null">

tx\_id,

</if>

<if test="recordId != null">

record\_id,

</if>

<if test="agentName != null">

agent\_name,

</if>

<if test="agentIdentityType != null">

agent\_identity\_type,

</if>

<if test="agentIdentityNo != null">

agent\_identity\_no,

</if>

<if test="agentPhotoUrl != null">

agent\_photo\_url,

</if>

<if test="agentPhotoHash != null">

agent\_photo\_hash,

</if>

<if test="agentVerifySpNo != null">

agent\_verify\_sp\_no,

</if>

<if test="agentVerifyOrderNo != null">

agent\_verify\_order\_no,

</if>

<if test="sealCode != null">

seal\_code,

</if>

<if test="sealUserCertNo != null">

seal\_user\_cert\_no,

</if>

<if test="originalFileHash != null">

original\_file\_hash,

</if>

<if test="stampingSign != null">

stamping\_sign,

</if>

<if test="stampingResultFileHash != null">

stamping\_result\_file\_hash,

</if>

<if test="stampingResult != null">

stamping\_result,

</if>

<if test="stampingMsg != null">

stamping\_msg,

</if>

<if test="stampingTime != null">

stamping\_time,

</if>

<if test="appId != null">

app\_id,

</if>

</trim>

<trim prefix="values (" suffix=")" suffixOverrides=",">

<if test="txId != null">

#{txId,jdbcType=VARCHAR},

</if>

<if test="recordId != null">

#{recordId,jdbcType=VARCHAR},

</if>

<if test="agentName != null">

#{agentName,jdbcType=VARCHAR},

</if>

<if test="agentIdentityType != null">

#{agentIdentityType,jdbcType=TINYINT},

</if>

<if test="agentIdentityNo != null">

#{agentIdentityNo,jdbcType=VARCHAR},

</if>

<if test="agentPhotoUrl != null">

#{agentPhotoUrl,jdbcType=VARCHAR},

</if>

<if test="agentPhotoHash != null">

#{agentPhotoHash,jdbcType=VARCHAR},

</if>

<if test="agentVerifySpNo != null">

#{agentVerifySpNo,jdbcType=VARCHAR},

</if>

<if test="agentVerifyOrderNo != null">

#{agentVerifyOrderNo,jdbcType=VARCHAR},

</if>

<if test="sealCode != null">

#{sealCode,jdbcType=VARCHAR},

</if>

<if test="sealUserCertNo != null">

#{sealUserCertNo,jdbcType=VARCHAR},

</if>

<if test="originalFileHash != null">

#{originalFileHash,jdbcType=VARCHAR},

</if>

<if test="stampingSign != null">

#{stampingSign,jdbcType=VARCHAR},

</if>

<if test="stampingResultFileHash != null">

#{stampingResultFileHash,jdbcType=VARCHAR},

</if>

<if test="stampingResult != null">

#{stampingResult,jdbcType=TINYINT},

</if>

<if test="stampingMsg != null">

#{stampingMsg,jdbcType=VARCHAR},

</if>

<if test="stampingTime != null">

#{stampingTime,jdbcType=BIGINT},

</if>

<if test="appId != null">

#{appId,jdbcType=VARCHAR},

</if>

</trim>

</insert>

agent\_name = #{agentName,jdbcType=VARCHAR},

agent\_identity\_type = #{agentIdentityType,jdbcType=TINYINT},

agent\_identity\_no = #{agentIdentityNo,jdbcType=VARCHAR},

agent\_photo\_url = #{agentPhotoUrl,jdbcType=VARCHAR},

agent\_photo\_hash = #{agentPhotoHash,jdbcType=VARCHAR},

agent\_verify\_sp\_no = #{agentVerifySpNo,jdbcType=VARCHAR},

agent\_verify\_order\_no = #{agentVerifyOrderNo,jdbcType=VARCHAR},

seal\_code = #{sealCode,jdbcType=VARCHAR},

seal\_user\_cert\_no = #{sealUserCertNo,jdbcType=VARCHAR},

original\_file\_hash = #{originalFileHash,jdbcType=VARCHAR},

stamping\_sign = #{stampingSign,jdbcType=VARCHAR},

stamping\_result\_file\_hash = #{stampingResultFileHash,jdbcType=VARCHAR},

stamping\_result = #{stampingResult,jdbcType=TINYINT},

stamping\_msg = #{stampingMsg,jdbcType=VARCHAR},

stamping\_time = #{stampingTime,jdbcType=BIGINT},

app\_id = #{appId,jdbcType=VARCHAR}

where tx\_id = #{txId,jdbcType=VARCHAR}

</update>

</mapper>

<?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="cn.g4b.fm.dao.szg.SzgHksecurityOrderMapper">

<resultMap id="BaseResultMap" type="cn.g4b.fm.model.szg.SzgHksecurityOrder">

<id column="tx\_id" jdbcType="VARCHAR" property="txId" />

<result column="order\_no" jdbcType="VARCHAR" property="orderNo" />

<result column="contract\_hash" jdbcType="VARCHAR" property="contractHash" />

<result column="app\_id" jdbcType="VARCHAR" property="appId" />

<result column="create\_time" jdbcType="TIMESTAMP" property="createTime" />

</resultMap>

<sql id="Base\_Column\_List">

tx\_id, order\_no, contract\_hash, app\_id, create\_time

</sql>

<select id="selectByPrimaryKey" parameterType="java.lang.String" resultMap="BaseResultMap">

select

<include refid="Base\_Column\_List" />

from szg\_hksecurity\_order

where tx\_id = #{txId,jdbcType=VARCHAR}

</select>

<delete id="deleteByPrimaryKey" parameterType="java.lang.String">

delete from szg\_hksecurity\_order

where tx\_id = #{txId,jdbcType=VARCHAR}

</delete>

<insert id="insert" parameterType="cn.g4b.fm.model.szg.SzgHksecurityOrder">

insert into szg\_hksecurity\_order (tx\_id, order\_no, contract\_hash,

app\_id, create\_time)

values (#{txId,jdbcType=VARCHAR}, #{orderNo,jdbcType=VARCHAR}, #{contractHash,jdbcType=VARCHAR},

#{appId,jdbcType=VARCHAR}, #{createTime,jdbcType=TIMESTAMP})

</insert>

<insert id="insertSelective" parameterType="cn.g4b.fm.model.szg.SzgHksecurityOrder">

insert into szg\_hksecurity\_order

<trim prefix="(" suffix=")" suffixOverrides=",">

<if test="txId != null">

tx\_id,

</if>

<if test="orderNo != null">

order\_no,

</if>

<if test="contractHash != null">

contract\_hash,

</if>

<if test="appId != null">

app\_id,

</if>

<if test="createTime != null">

create\_time,

</if>

</trim>

<trim prefix="values (" suffix=")" suffixOverrides=",">

<if test="txId != null">

#{txId,jdbcType=VARCHAR},

</if>

<if test="orderNo != null">

#{orderNo,jdbcType=VARCHAR},

</if>

<if test="contractHash != null">

#{contractHash,jdbcType=VARCHAR},

</if>

<if test="appId != null">

#{appId,jdbcType=VARCHAR},

</if>

<if test="createTime != null">

#{createTime,jdbcType=TIMESTAMP},

</if>

</trim>

</insert>

<update id="updateByPrimaryKeySelective" parameterType="cn.g4b.fm.model.szg.SzgHksecurityOrder">

update szg\_hksecurity\_order

<set>

<if test="orderNo != null">

order\_no = #{orderNo,jdbcType=VARCHAR},

</if>

<if test="contractHash != null">

contract\_hash = #{contractHash,jdbcType=VARCHAR},

</if>

<if test="appId != null">

app\_id = #{appId,jdbcType=VARCHAR},

</if>

<if test="createTime != null">

create\_time = #{createTime,jdbcType=TIMESTAMP},

</if>

</set>

where tx\_id = #{txId,jdbcType=VARCHAR}

</update>

<update id="updateByPrimaryKey" parameterType="cn.g4b.fm.model.szg.SzgHksecurityOrder">

update szg\_hksecurity\_order

set order\_no = #{orderNo,jdbcType=VARCHAR},

contract\_hash = #{contractHash,jdbcType=VARCHAR},

app\_id = #{appId,jdbcType=VARCHAR},

create\_time = #{createTime,jdbcType=TIMESTAMP}

where tx\_id = #{txId,jdbcType=VARCHAR}

</update>

</mapper>

<?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="cn.g4b.fm.dao.szg.SzgOrderEventLogMapper">

<resultMap id="BaseResultMap" type="cn.g4b.fm.model.szg.SzgOrderEventLog">

<id column="id" jdbcType="INTEGER" property="id" />

<result column="order\_id" jdbcType="VARCHAR" property="orderId" />

<result column="time" jdbcType="TIMESTAMP" property="time" />

<result column="event\_name" jdbcType="VARCHAR" property="eventName" />

<result column="tx\_id" jdbcType="VARCHAR" property="txId" />

</resultMap>

<sql id="Base\_Column\_List">

id, order\_id, `time`, event\_name, tx\_id

</sql>

<select id="selectByPrimaryKey" parameterType="java.lang.Integer" resultMap="BaseResultMap">

select

<include refid="Base\_Column\_List" />

from szg\_order\_event\_log

where id = #{id,jdbcType=INTEGER}

</select>

<select id="selectByHeaderKey" parameterType="java.lang.String" resultMap="BaseResultMap">

select

<include refid="Base\_Column\_List" />

from szg\_order\_event\_log

where header\_key = #{headerKey,jdbcType=VARCHAR}

limit 1

</select>

<select id="getListByOrderId" parameterType="java.lang.String" resultMap="BaseResultMap">

select

<include refid="Base\_Column\_List" />

from szg\_order\_event\_log

where order\_id = #{orderId,jdbcType=VARCHAR}

order by time asc

</select>

<delete id="deleteByPrimaryKey" parameterType="java.lang.Integer">

delete from szg\_order\_event\_log

where id = #{id,jdbcType=INTEGER}

</delete>

<insert id="insert" parameterType="cn.g4b.fm.model.szg.SzgOrderEventLog">

insert into szg\_order\_event\_log (id, order\_id, `time`,

event\_name, tx\_id

)

values (#{id,jdbcType=INTEGER}, #{orderId,jdbcType=VARCHAR}, #{time,jdbcType=TIMESTAMP},

#{eventName,jdbcType=VARCHAR}, #{txId,jdbcType=VARCHAR}

)

</insert>

<insert id="insertSelective" parameterType="cn.g4b.fm.model.szg.SzgOrderEventLog">

insert into szg\_order\_event\_log

<trim prefix="(" suffix=")" suffixOverrides=",">

<if test="id != null">

id,

</if>

<if test="orderId != null">

order\_id,

</if>

<if test="time != null">

`time`,

</if>

<if test="eventName != null">

event\_name,

</if>

<if test="txId != null">

tx\_id,

</if>

</trim>

<trim prefix="values (" suffix=")" suffixOverrides=",">

<if test="id != null">

#{id,jdbcType=INTEGER},

</if>

<if test="orderId != null">

#{orderId,jdbcType=VARCHAR},

</if>

<if test="time != null">

#{time,jdbcType=TIMESTAMP},

</if>

<if test="eventName != null">

#{eventName,jdbcType=VARCHAR},

</if>

<if test="txId != null">

#{txId,jdbcType=VARCHAR},

</if>

</trim>

</insert>

<update id="updateByPrimaryKeySelective" parameterType="cn.g4b.fm.model.szg.SzgOrderEventLog">

update szg\_order\_event\_log

<set>

<if test="orderId != null">

order\_id = #{orderId,jdbcType=VARCHAR},

</if>

<if test="time != null">

`time` = #{time,jdbcType=TIMESTAMP},

</if>

<if test="eventName != null">

event\_name = #{eventName,jdbcType=VARCHAR},

</if>

<if test="txId != null">

tx\_id = #{txId,jdbcType=VARCHAR},

</if>

</set>

where id = #{id,jdbcType=INTEGER}

</update>

<update id="updateByPrimaryKey" parameterType="cn.g4b.fm.model.szg.SzgOrderEventLog">

update szg\_order\_event\_log

set order\_id = #{orderId,jdbcType=VARCHAR},

`time` = #{time,jdbcType=TIMESTAMP},

event\_name = #{eventName,jdbcType=VARCHAR},

tx\_id = #{txId,jdbcType=VARCHAR}

where id = #{id,jdbcType=INTEGER}

</update>

</mapper>

package cn.g4b.fm.model.szg;

import cn.g4b.fm.model.sys.CbApp;

import com.fasterxml.jackson.annotation.JsonFormat;

import java.io.Serializable;

import java.util.Date;

/\*\*

\* @author

\* 业务方接入扩展信息

主要是为了扩展市场部的功能

\*/

public class CbAppExtendInfo extends CbApp implements Serializable {

private String appId;

private String depName;

private String entCode;

private String linkAddress;

private String linkPerson;

private String depPost;

private String linkPhone;

private String email;

/\*\*

\* 机构性质（单选）

0:政府机关

1:事业单位

2:国企

3:机构团体

4:其他

\*/

private Integer entNature;

/\*\*

\* 网络性质（单选）

0:广州市政务网

1:互联网

2:其他

\*/

private Integer netNature;

private String ip;

/\*\*

\* 应用需求（多选）

0:申请电子印章服务

1:签章服务

2:其他

\*/

private String appDemand;

private String userNumber;

private String concurrency;

@JsonFormat(pattern = "yyyy-MM-dd HH:mm:ss",timezone = "GMT+08")

private Date auditTime;

private String remark;

private static final long serialVersionUID = 1L;

public Date getAuditTime() {

return auditTime;

}

public void setAuditTime(Date auditTime) {

this.auditTime = auditTime;

}

public String getAppId() {

return appId;

}

public void setAppId(String appId) {

this.appId = appId;

}

public String getDepName() {

return depName;

}

public void setDepName(String depName) {

this.depName = depName;

}

public String getEntCode() {

return entCode;

}

public void setEntCode(String entCode) {

this.entCode = entCode;

}

public String getLinkAddress() {

return linkAddress;

}

public void setLinkAddress(String linkAddress) {

this.linkAddress = linkAddress;

}

public String getLinkPerson() {

return linkPerson;

}

public void setLinkPerson(String linkPerson) {

this.linkPerson = linkPerson;

}

public String getDepPost() {

return depPost;

}

public void setDepPost(String depPost) {

this.depPost = depPost;

}

public String getLinkPhone() {

return linkPhone;

}

public void setLinkPhone(String linkPhone) {

this.linkPhone = linkPhone;

}

public String getEmail() {

return email;

}

public void setEmail(String email) {

this.email = email;

}

public Integer getEntNature() {

return entNature;

}

public void setEntNature(Integer entNature) {

this.entNature = entNature;

}

public Integer getNetNature() {

return netNature;

}

public void setNetNature(Integer netNature) {

this.netNature = netNature;

}

public String getIp() {

return ip;

}

public void setIp(String ip) {

this.ip = ip;

}

public String getAppDemand() {

return appDemand;

}

public void setAppDemand(String appDemand) {

this.appDemand = appDemand;

}

public String getUserNumber() {

return userNumber;

}

public void setUserNumber(String userNumber) {

this.userNumber = userNumber;

}

public String getConcurrency() {

return concurrency;

}

public void setConcurrency(String concurrency) {

this.concurrency = concurrency;

}

public String getRemark() {

return remark;

}

public void setRemark(String remark) {

this.remark = remark;

}

@Override

public boolean equals(Object that) {

if (this == that) {

return true;

}

if (that == null) {

return false;

}

if (getClass() != that.getClass()) {

return false;

}

CbAppExtendInfo other = (CbAppExtendInfo) that;

return (this.getAppId() == null ? other.getAppId() == null : this.getAppId().equals(other.getAppId()))

&& (this.getDepName() == null ? other.getDepName() == null : this.getDepName().equals(other.getDepName()))

&& (this.getEntCode() == null ? other.getEntCode() == null : this.getEntCode().equals(other.getEntCode()))

&& (this.getLinkAddress() == null ? other.getLinkAddress() == null : this.getLinkAddress().equals(other.getLinkAddress()))

&& (this.getLinkPerson() == null ? other.getLinkPerson() == null : this.getLinkPerson().equals(other.getLinkPerson()))

&& (this.getDepPost() == null ? other.getDepPost() == null : this.getDepPost().equals(other.getDepPost()))

&& (this.getLinkPhone() == null ? other.getLinkPhone() == null : this.getLinkPhone().equals(other.getLinkPhone()))

&& (this.getEmail() == null ? other.getEmail() == null : this.getEmail().equals(other.getEmail()))

&& (this.getEntNature() == null ? other.getEntNature() == null : this.getEntNature().equals(other.getEntNature()))

&& (this.getNetNature() == null ? other.getNetNature() == null : this.getNetNature().equals(other.getNetNature()))

&& (this.getIp() == null ? other.getIp() == null : this.getIp().equals(other.getIp()))

&& (this.getAppDemand() == null ? other.getAppDemand() == null : this.getAppDemand().equals(other.getAppDemand()))

&& (this.getUserNumber() == null ? other.getUserNumber() == null : this.getUserNumber().equals(other.getUserNumber()))

&& (this.getConcurrency() == null ? other.getConcurrency() == null : this.getConcurrency().equals(other.getConcurrency()))

&& (this.getRemark() == null ? other.getRemark() == null : this.getRemark().equals(other.getRemark()));

}

@Override

public int hashCode() {

final int prime = 31;

int result = 1;

result = prime \* result + ((getAppId() == null) ? 0 : getAppId().hashCode());

result = prime \* result + ((getDepName() == null) ? 0 : getDepName().hashCode());

result = prime \* result + ((getEntCode() == null) ? 0 : getEntCode().hashCode());

result = prime \* result + ((getLinkAddress() == null) ? 0 : getLinkAddress().hashCode());

result = prime \* result + ((getLinkPerson() == null) ? 0 : getLinkPerson().hashCode());

result = prime \* result + ((getDepPost() == null) ? 0 : getDepPost().hashCode());

result = prime \* result + ((getLinkPhone() == null) ? 0 : getLinkPhone().hashCode());

result = prime \* result + ((getEmail() == null) ? 0 : getEmail().hashCode());

result = prime \* result + ((getEntNature() == null) ? 0 : getEntNature().hashCode());

result = prime \* result + ((getNetNature() == null) ? 0 : getNetNature().hashCode());

result = prime \* result + ((getIp() == null) ? 0 : getIp().hashCode());

result = prime \* result + ((getAppDemand() == null) ? 0 : getAppDemand().hashCode());

result = prime \* result + ((getUserNumber() == null) ? 0 : getUserNumber().hashCode());

result = prime \* result + ((getConcurrency() == null) ? 0 : getConcurrency().hashCode());

result = prime \* result + ((getRemark() == null) ? 0 : getRemark().hashCode());

return result;

}

@Override

public String toString() {

StringBuilder sb = new StringBuilder();

sb.append(getClass().getSimpleName());

sb.append(" [");

sb.append("Hash = ").append(hashCode());

sb.append(", appId=").append(appId);

sb.append(", depName=").append(depName);

sb.append(", entCode=").append(entCode);

sb.append(", linkAddress=").append(linkAddress);

sb.append(", linkPerson=").append(linkPerson);

sb.append(", depPost=").append(depPost);

sb.append(", linkPhone=").append(linkPhone);

sb.append(", email=").append(email);

sb.append(", entNature=").append(entNature);

sb.append(", netNature=").append(netNature);

sb.append(", ip=").append(ip);

sb.append(", appDemand=").append(appDemand);

sb.append(", userNumber=").append(userNumber);

sb.append(", concurrency=").append(concurrency);

sb.append(", remark=").append(remark);

sb.append(", serialVersionUID=").append(serialVersionUID);

sb.append("]");

return sb.toString();

}

}

package cn.g4b.fm.model.szg;

import com.fasterxml.jackson.annotation.JsonFormat;

import java.io.Serializable;

import java.util.Date;

/\*\*

\* @author

\*

\*/

public class ServiceProvider implements Serializable {

private Integer id;

/\*\*

\* 服务商编号，自定义不允许重复

\*/

private String providerNo;

/\*\*

\* 服务商名称

\*/

private String providerName;

/\*\*

\* 证书ID

\*/

private String certId;

/\*\*

\* 证书DN项

\*/

private String certDn;

/\*\*

\* 用户证书标识

\*/

private String userIdentity;

/\*\*

\* 证书申请实体

\*/

private String applicant;

/\*\*

\* 证件号，当申请者为企业是为0

\*/

private String identityNum;

private String csrData;

private String csrUrl;

/\*\*

\* 扩展字段，目前没用

\*/

private Integer providerLevel = 1;

/\*\*

\* 1：ca厂商 2：印章申请单位

\*/

private Integer providerType;

/\*\*

\* 1：启用 2：暂停 3：终止

\*/

private Byte state;

@JsonFormat(pattern = "yyyy-MM-dd HH:mm:ss",timezone = "GMT+08")

private Date createTime;

@JsonFormat(pattern = "yyyy-MM-dd HH:mm:ss",timezone = "GMT+08")

private Date updateTime;

/\*\*

\* sp所在区域 0 大陆 1 香港， 默认为0

\*/

private String spRegion;

/\*\*

\* 服务商类型

\* 1 印章服务商

\* 2 证书服务商

\* 默认值为1

\*/

private Integer spType;

/\*\*

\* fabric返回的交易ID

\*/

private String txId;

private String groupName;//所属fabric组织

private String orgId;//政府机构部门标识代码

private String orgSysCode;//政府机构业务系统编号

private String clientCert;

private String callbackUrl;//新订单通知回调地址

private Integer sealFrom;//0 可以申请新印章和提交存量印章 1 只能申请新印章 2 只能提交存量印章，默认为0

private static final long serialVersionUID = 1L;

public Integer getId() {

return id;

}

public void setId(Integer id) {

this.id = id;

}

public String getProviderNo() {

return providerNo;

}

public void setProviderNo(String providerNo) {

this.providerNo = providerNo;

}

public String getProviderName() {

return providerName;

}

public void setProviderName(String providerName) {

this.providerName = providerName;

}

public String getCertId() {

return certId;

}

public void setCertId(String certId) {

this.certId = certId;

}

public String getCertDn() {

return certDn;

}

public void setCertDn(String certDn) {

this.certDn = certDn;

}

public String getUserIdentity() {

return userIdentity;

}

public void setUserIdentity(String userIdentity) {

this.userIdentity = userIdentity;

}

public String getApplicant() {

return applicant;

}

public void setApplicant(String applicant) {

this.applicant = applicant;

}

public String getIdentityNum() {

return identityNum;

}

public void setIdentityNum(String identityNum) {

this.identityNum = identityNum;

}

public String getCsrData() {

return csrData;

}

public void setCsrData(String csrData) {

this.csrData = csrData;

}

public Integer getProviderLevel() {

return providerLevel;

}

public void setProviderLevel(Integer providerLevel) {

this.providerLevel = providerLevel;

}

public Integer getProviderType() {

return providerType;

}

public void setProviderType(Integer providerType) {

this.providerType = providerType;

}

public Byte getState() {

return state;

}

public void setState(Byte state) {

this.state = state;

}

public Date getCreateTime() {

return createTime;

}

public void setCreateTime(Date createTime) {

this.createTime = createTime;

}

public Date getUpdateTime() {

return updateTime;

}

public void setUpdateTime(Date updateTime) {

this.updateTime = updateTime;

}

public String getTxId() {

return txId;

}

public void setTxId(String txId) {

this.txId = txId;

}

public String getSpRegion() {

return spRegion;

}

public void setSpRegion(String spRegion) {

this.spRegion = spRegion;

}

public String getClientCert() {

return clientCert;

}

public void setClientCert(String clientCert) {

this.clientCert = clientCert;

}

public String getCsrUrl() {

return csrUrl;

}

public void setCsrUrl(String csrUrl) {

this.csrUrl = csrUrl;

}

public Integer getSpType() {

return spType;

}

public void setSpType(Integer spType) {

this.spType = spType;

}

@Override

public boolean equals(Object that) {

if (this == that) {

return true;

}

if (that == null) {

return false;

}

if (getClass() != that.getClass()) {

return false;

}

ServiceProvider other = (ServiceProvider) that;

return (this.getId() == null ? other.getId() == null : this.getId().equals(other.getId()))

&& (this.getProviderNo() == null ? other.getProviderNo() == null : this.getProviderNo().equals(other.getProviderNo()))

&& (this.getProviderName() == null ? other.getProviderName() == null : this.getProviderName().equals(other.getProviderName()))

&& (this.getCertId() == null ? other.getCertId() == null : this.getCertId().equals(other.getCertId()))

&& (this.getCertDn() == null ? other.getCertDn() == null : this.getCertDn().equals(other.getCertDn()))

&& (this.getUserIdentity() == null ? other.getUserIdentity() == null : this.getUserIdentity().equals(other.getUserIdentity()))

&& (this.getApplicant() == null ? other.getApplicant() == null : this.getApplicant().equals(other.getApplicant()))

&& (this.getIdentityNum() == null ? other.getIdentityNum() == null : this.getIdentityNum().equals(other.getIdentityNum()))

&& (this.getCsrData() == null ? other.getCsrData() == null : this.getCsrData().equals(other.getCsrData()))

&& (this.getProviderLevel() == null ? other.getProviderLevel() == null : this.getProviderLevel().equals(other.getProviderLevel()))

&& (this.getProviderType() == null ? other.getProviderType() == null : this.getProviderType().equals(other.getProviderType()))

&& (this.getState() == null ? other.getState() == null : this.getState().equals(other.getState()))

&& (this.getCreateTime() == null ? other.getCreateTime() == null : this.getCreateTime().equals(other.getCreateTime()))

&& (this.getUpdateTime() == null ? other.getUpdateTime() == null : this.getUpdateTime().equals(other.getUpdateTime()))

&& (this.getTxId() == null ? other.getTxId() == null : this.getTxId().equals(other.getTxId()));

}

@Override

public int hashCode() {

final int prime = 31;

int result = 1;

result = prime \* result + ((getId() == null) ? 0 : getId().hashCode());

result = prime \* result + ((getProviderNo() == null) ? 0 : getProviderNo().hashCode());

result = prime \* result + ((getProviderName() == null) ? 0 : getProviderName().hashCode());

result = prime \* result + ((getCertId() == null) ? 0 : getCertId().hashCode());

result = prime \* result + ((getCertDn() == null) ? 0 : getCertDn().hashCode());

result = prime \* result + ((getUserIdentity() == null) ? 0 : getUserIdentity().hashCode());

result = prime \* result + ((getApplicant() == null) ? 0 : getApplicant().hashCode());

result = prime \* result + ((getIdentityNum() == null) ? 0 : getIdentityNum().hashCode());

result = prime \* result + ((getCsrData() == null) ? 0 : getCsrData().hashCode());

result = prime \* result + ((getProviderLevel() == null) ? 0 : getProviderLevel().hashCode());

result = prime \* result + ((getProviderType() == null) ? 0 : getProviderType().hashCode());

result = prime \* result + ((getState() == null) ? 0 : getState().hashCode());

result = prime \* result + ((getCreateTime() == null) ? 0 : getCreateTime().hashCode());

result = prime \* result + ((getUpdateTime() == null) ? 0 : getUpdateTime().hashCode());

result = prime \* result + ((getTxId() == null) ? 0 : getTxId().hashCode());

return result;

}

@Override

public String toString() {

StringBuilder sb = new StringBuilder();

sb.append(getClass().getSimpleName());

sb.append(" [");

sb.append("Hash = ").append(hashCode());

sb.append(", id=").append(id);

sb.append(", providerNo=").append(providerNo);

sb.append(", providerName=").append(providerName);

sb.append(", certId=").append(certId);

sb.append(", certDn=").append(certDn);

sb.append(", userIdentity=").append(userIdentity);

sb.append(", applicant=").append(applicant);

sb.append(", identityNum=").append(identityNum);

sb.append(", csrData=").append(csrData);

sb.append(", providerLevel=").append(providerLevel);

sb.append(", providerType=").append(providerType);

sb.append(", state=").append(state);

sb.append(", createTime=").append(createTime);

sb.append(", updateTime=").append(updateTime);

sb.append(", txId=").append(txId);

sb.append(", serialVersionUID=").append(serialVersionUID);

sb.append("]");

return sb.toString();

}

public String getGroupName() {

return groupName;

}

public void setGroupName(String groupName) {

this.groupName = groupName;

}

public String getOrgId() {

return orgId;

}

public void setOrgId(String orgId) {

this.orgId = orgId;

}

public String getOrgSysCode() {

return orgSysCode;

}

public void setOrgSysCode(String orgSysCode) {

this.orgSysCode = orgSysCode;

}

public String getCallbackUrl() {

return callbackUrl;

}

public void setCallbackUrl(String callbackUrl) {

this.callbackUrl = callbackUrl;

}

public Integer getSealFrom() {

return sealFrom;

}

public void setSealFrom(Integer sealFrom) {

this.sealFrom = sealFrom;

}

}

package cn.g4b.fm.model.szg;

import java.io.Serializable;

/\*\*

\* @author

\* 配置系统接口

\*/

public class SzgApi implements Serializable {

private Integer id;

private String apiName;

private String apiDesc;

private String apiUrl;

/\*\*

\* get post put delete

\*/

private String requestType;

/\*\*

\* 0：启用

1：禁用

\*/

private Integer status;

/\*\*

\* 0：全部对象

1：业务方

2：服务方

\*/

private Byte useObj;

private static final long serialVersionUID = 1L;

public Integer getId() {

return id;

}

public void setId(Integer id) {

this.id = id;

}

public String getApiName() {

return apiName;

}

public void setApiName(String apiName) {

this.apiName = apiName;

}

public String getApiDesc() {

return apiDesc;

}

public void setApiDesc(String apiDesc) {

this.apiDesc = apiDesc;

}

public String getApiUrl() {

return apiUrl;

}

public void setApiUrl(String apiUrl) {

this.apiUrl = apiUrl;

}

public String getRequestType() {

return requestType;

}

public void setRequestType(String requestType) {

this.requestType = requestType;

}

public Integer getStatus() {

return status;

}

public void setStatus(Integer status) {

this.status = status;

}

public Byte getUseObj() {

return useObj;

}

public void setUseObj(Byte useObj) {

this.useObj = useObj;

}

@Override

public boolean equals(Object that) {

if (this == that) {

return true;

}

if (that == null) {

return false;

}

if (getClass() != that.getClass()) {

return false;

}

SzgApi other = (SzgApi) that;

return (this.getId() == null ? other.getId() == null : this.getId().equals(other.getId()))

&& (this.getApiName() == null ? other.getApiName() == null : this.getApiName().equals(other.getApiName()))

&& (this.getApiDesc() == null ? other.getApiDesc() == null : this.getApiDesc().equals(other.getApiDesc()))

&& (this.getApiUrl() == null ? other.getApiUrl() == null : this.getApiUrl().equals(other.getApiUrl()))

&& (this.getRequestType() == null ? other.getRequestType() == null : this.getRequestType().equals(other.getRequestType()))

&& (this.getStatus() == null ? other.getStatus() == null : this.getStatus().equals(other.getStatus()))

&& (this.getUseObj() == null ? other.getUseObj() == null : this.getUseObj().equals(other.getUseObj()));

}

@Override

public int hashCode() {

final int prime = 31;

int result = 1;

result = prime \* result + ((getId() == null) ? 0 : getId().hashCode());

result = prime \* result + ((getApiName() == null) ? 0 : getApiName().hashCode());

result = prime \* result + ((getApiDesc() == null) ? 0 : getApiDesc().hashCode());

result = prime \* result + ((getApiUrl() == null) ? 0 : getApiUrl().hashCode());

result = prime \* result + ((getRequestType() == null) ? 0 : getRequestType().hashCode());

result = prime \* result + ((getStatus() == null) ? 0 : getStatus().hashCode());

result = prime \* result + ((getUseObj() == null) ? 0 : getUseObj().hashCode());

return result;

}

@Override

public String toString() {

StringBuilder sb = new StringBuilder();

sb.append(getClass().getSimpleName());

sb.append(" [");

sb.append("Hash = ").append(hashCode());

sb.append(", id=").append(id);

sb.append(", apiName=").append(apiName);

sb.append(", apiDesc=").append(apiDesc);

sb.append(", apiUrl=").append(apiUrl);

sb.append(", requestType=").append(requestType);

sb.append(", status=").append(status);

sb.append(", useObj=").append(useObj);

sb.append(", serialVersionUID=").append(serialVersionUID);

sb.append("]");

return sb.toString();

}

}

package cn.g4b.fm.model.szg;

import java.io.Serializable;

/\*\*

\* @author

\* 配置系统接口

\*/

public class SzgApiWithBLOBs extends SzgApi implements Serializable {

private String requestHeader;

private String requestBody;

private String responseBody;

private static final long serialVersionUID = 1L;

public String getRequestHeader() {

return requestHeader;

}

public void setRequestHeader(String requestHeader) {

this.requestHeader = requestHeader;

}

public String getRequestBody() {

return requestBody;

}

public void setRequestBody(String requestBody) {

this.requestBody = requestBody;

}

public String getResponseBody() {

return responseBody;

}

public void setResponseBody(String responseBody) {

this.responseBody = responseBody;

}

@Override

public boolean equals(Object that) {

if (this == that) {

return true;

}

if (that == null) {

return false;

}

if (getClass() != that.getClass()) {

return false;

}

SzgApiWithBLOBs other = (SzgApiWithBLOBs) that;

return (this.getId() == null ? other.getId() == null : this.getId().equals(other.getId()))

&& (this.getApiName() == null ? other.getApiName() == null : this.getApiName().equals(other.getApiName()))

&& (this.getApiDesc() == null ? other.getApiDesc() == null : this.getApiDesc().equals(other.getApiDesc()))

&& (this.getApiUrl() == null ? other.getApiUrl() == null : this.getApiUrl().equals(other.getApiUrl()))

&& (this.getRequestType() == null ? other.getRequestType() == null : this.getRequestType().equals(other.getRequestType()))

&& (this.getStatus() == null ? other.getStatus() == null : this.getStatus().equals(other.getStatus()))

&& (this.getUseObj() == null ? other.getUseObj() == null : this.getUseObj().equals(other.getUseObj()))

&& (this.getRequestHeader() == null ? other.getRequestHeader() == null : this.getRequestHeader().equals(other.getRequestHeader()))

&& (this.getRequestBody() == null ? other.getRequestBody() == null : this.getRequestBody().equals(other.getRequestBody()))

&& (this.getResponseBody() == null ? other.getResponseBody() == null : this.getResponseBody().equals(other.getResponseBody()));

}

@Override

public int hashCode() {

final int prime = 31;

int result = 1;

result = prime \* result + ((getId() == null) ? 0 : getId().hashCode());

result = prime \* result + ((getApiName() == null) ? 0 : getApiName().hashCode());

result = prime \* result + ((getApiDesc() == null) ? 0 : getApiDesc().hashCode());

result = prime \* result + ((getApiUrl() == null) ? 0 : getApiUrl().hashCode());

result = prime \* result + ((getRequestType() == null) ? 0 : getRequestType().hashCode());

result = prime \* result + ((getStatus() == null) ? 0 : getStatus().hashCode());

result = prime \* result + ((getUseObj() == null) ? 0 : getUseObj().hashCode());

result = prime \* result + ((getRequestHeader() == null) ? 0 : getRequestHeader().hashCode());

result = prime \* result + ((getRequestBody() == null) ? 0 : getRequestBody().hashCode());

result = prime \* result + ((getResponseBody() == null) ? 0 : getResponseBody().hashCode());

return result;

}

@Override

public String toString() {

StringBuilder sb = new StringBuilder();

sb.append(getClass().getSimpleName());

sb.append(" [");

sb.append("Hash = ").append(hashCode());

sb.append(", requestHeader=").append(requestHeader);

sb.append(", requestBody=").append(requestBody);

sb.append(", responseBody=").append(responseBody);

sb.append(", serialVersionUID=").append(serialVersionUID);

sb.append("]");

return sb.toString();

}

}

package cn.g4b.fm.model.szg;

import java.io.Serializable;

/\*\*

\* @author

\* 应用相应api权限中间表

\*/

public class SzgAppApi implements Serializable {

private String appId;

private Integer apiId;

private static final long serialVersionUID = 1L;

public SzgAppApi(Integer apiId, String appId) {

this.apiId = apiId;

this.appId = appId;

}

public String getAppId() {

return appId;

}

public void setAppId(String appId) {

this.appId = appId;

}

public Integer getApiId() {

return apiId;

}

public void setApiId(Integer apiId) {

this.apiId = apiId;

}

@Override

public boolean equals(Object that) {

if (this == that) {

return true;

}

if (that == null) {

return false;

}

if (getClass() != that.getClass()) {

return false;

}

SzgAppApi other = (SzgAppApi) that;

return (this.getAppId() == null ? other.getAppId() == null : this.getAppId().equals(other.getAppId()))

&& (this.getApiId() == null ? other.getApiId() == null : this.getApiId().equals(other.getApiId()));

}

@Override

public int hashCode() {

final int prime = 31;

int result = 1;

result = prime \* result + ((getAppId() == null) ? 0 : getAppId().hashCode());

result = prime \* result + ((getApiId() == null) ? 0 : getApiId().hashCode());

return result;

}

@Override

public String toString() {

StringBuilder sb = new StringBuilder();

sb.append(getClass().getSimpleName());

sb.append(" [");

sb.append("Hash = ").append(hashCode());

sb.append(", appId=").append(appId);

sb.append(", apiId=").append(apiId);

sb.append(", serialVersionUID=").append(serialVersionUID);

sb.append("]");

return sb.toString();

}

}

package main

import (

"encoding/json"

"fmt"

"errors"

"time"

"bytes"

"strconv"

"encoding/pem"

"strings"

"runtime"

"crypto/x509"

"crypto/md5"

"github.com/hyperledger/fabric/core/chaincode/shim"

pb "github.com/hyperledger/fabric/protos/peer"

)

func (t \*SimpleChaincode) AddLogo(stub shim.ChaincodeStubInterface, args []string) pb.Response {

if len(args) != 1 {

Msg := "{\"Result\": false, \"Msg\": \"Incorrect number of arguments. Expecting 1\", \"OrderId\",\"\"}"

return shim.Error(Msg)

}

var newInfo SealLogoInfo

//json str 转struct

if err := json.Unmarshal([]byte(args[0]), &newInfo); err != nil {

\_, \_, line, \_ := runtime.Caller(0)

errMsg := fmt.Sprintf("Invalid input arg: %s", args[0])

msg := fmt.Sprintf("{\"Result\": false, \"Msg\": \"%s. line: %d\"}", errMsg, line)

return shim.Error(msg)

}

if err := AddSealLogoPreHandle(stub, &newInfo); err != nil {

msg := fmt.Sprintf("{\"Result\": false, \"Msg\": \"%s\"}", err.Error())

return shim.Error(msg)

}

// 先检查记录是否已存在，如果是，添加更改记录

key := SealLogoKeyPrefix + newInfo.EntCode + "\_" + strconv.Itoa(int(newInfo.SealType)) + "\_" + newInfo.SealLogoRecordCodeHash

bytes, err3 := stub.GetState(key)

if err3 != nil {

\_, \_, line, \_ := runtime.Caller(0)

msg := fmt.Sprintf("{\"Result\": false, \"Msg\": \"%s\". line: %d\"}", err3.Error(), line)

return shim.Error(msg)

}

if bytes != nil { // 原来有记录

var oldInfo SealLogoInfo

if err := json.Unmarshal(bytes, &oldInfo); err != nil {

\_, \_, line, \_ := runtime.Caller(0)

msg := fmt.Sprintf("error：%s. line: %d", err.Error(), line)

return shim.Error(msg)

}

var operationType uint = 1 //更新文件

err := SealLogoChangeLogHandle(stub, key, operationType, &oldInfo, &newInfo)

if err != nil {

\_, \_, line, \_ := runtime.Caller(0)

msg := fmt.Sprintf("{\"Result\": false, \"Msg\": \"error: %s, line: %d\"}", err.Error(), line)

return shim.Error(msg)

}

}

bytes2, err2 := json.Marshal(newInfo);

if err2 != nil {

\_, \_, line, \_ := runtime.Caller(0)

msg := fmt.Sprintf("{\"Result\": false, \"Msg\": \"error: %s, line: %d\"}", err2.Error(), line)

return shim.Error(msg)

}

// 写db

err := stub.PutState(key, bytes2)

if err != nil {

\_, \_, line, \_ := runtime.Caller(0)

msg := fmt.Sprintf("{\"Result\": false, \"Msg\": \"error: %s, line: %d\"}", err.Error(), line)

return shim.Error(msg)

}

msg := fmt.Sprintf("{\"Result\": true, \"Msg\": \"200\"}")

return shim.Success([]byte(msg))

}

func AddSealLogoPreHandle(stub shim.ChaincodeStubInterface, info \*SealLogoInfo) error {

if info.EntName == "" {

\_, \_, line, \_ := runtime.Caller(0)

msg := fmt.Sprintf("10003：EntName参数值需不能为空. line: %d", line)

return errors.New(msg)

}

if info.EntType == "" {

\_, \_, line, \_ := runtime.Caller(0)

msg := fmt.Sprintf("10003：EntType参数值需不能为空. line: %d", line)

return errors.New(msg)

}

if info.EntCode == "" {

\_, \_, line, \_ := runtime.Caller(0)

msg := fmt.Sprintf("10003：EntCode参数值需不能为空. line: %d", line)

return errors.New(msg)

}

// 检查每个新印章的印章类型是否合法

if info.SealType < 1 || (info.SealType > 6 && info.SealType != 99) { //印章类型有：1 单位专用印章 2 财务专用章 3 税务专用章 4 合同专用章 5 法定代表人名章 6 名章 99 其他类型电子印章

\_, \_, line, \_ := runtime.Caller(0)

msg := fmt.Sprintf("40004：该印章类型不合法。印章类型：%d。 印章类型合法值范围 1-6和99. line: %d", info.SealType, line)

return errors.New(msg)

}

if info.SealLogoRecordCodeHash == "" {

\_, \_, line, \_ := runtime.Caller(0)

msg := fmt.Sprintf("10003：SealLogoRecordCodeHash参数值需不能为空. line: %d", line)

return errors.New(msg)

}

if info.SealLogoFileUrl == "" {

\_, \_, line, \_ := runtime.Caller(0)

msg := fmt.Sprintf("10003：SealLogoFileUrl参数值需不能为空. line: %d", line)

return errors.New(msg)

}

if info.SealLogoFileHash == "" {

\_, \_, line, \_ := runtime.Caller(0)

msg := fmt.Sprintf("10003：SealLogoFileHash参数值需不能为空. line: %d", line)

return errors.New(msg)

}

if info.SealLogoExpireDate <= 0 {

\_, \_, line, \_ := runtime.Caller(0)

msg := fmt.Sprintf("10003：SealLogoExpireDate参数值需大于0. line: %d", line)

return errors.New(msg)

}

if info.AppId == "" {

\_, \_, line, \_ := runtime.Caller(0)

msg := fmt.Sprintf("10003：AppId参数值需不能为空. line: %d", line)

return errors.New(msg)

}

info.SealLogoStatus = 1 // 印模状态 0 失效 1 有效

return nil

}

func SealLogoChangeLogHandle(stub shim.ChaincodeStubInterface, key string, operationType uint, oldInfo, newInfo \*SealLogoInfo) error {

// 添加更改终端用户状态记录

t1, err4 := stub.GetTxTimestamp()

if err4 != nil {

\_, \_, line, \_ := runtime.Caller(0)

msg := fmt.Sprintf("{\"Result\": false, \"Msg\": \"GetTxTimestamp error:%s, line: %d\"}", err4.Error(), line)

return errors.New(msg)

}

var log ChangeLog

switch(operationType){

case 1:

log.Operation = "更新文件"

break;

case 2:

log.Operation = "更新状态为失效"

\*newInfo = \*oldInfo

break

default:

\_, \_, line, \_ := runtime.Caller(0)

msg := fmt.Sprintf("{\"Result\": false, \"Msg\": \"error: wrong operationType%s, line: %d\"}",operationType, line)

return errors.New(msg)

}

log.OperTime = time.Unix(t1.Seconds, int64(t1.Nanos)).UnixNano() / 1e6 // unix时间戳，精确到毫秒。

log.TxId = stub.GetTxID()

log.OldFileUrl = oldInfo.SealLogoFileUrl

log.OldFileHash = oldInfo.SealLogoFileHash

oldInfo.ChangeList = append(oldInfo.ChangeList, log)

newInfo.ChangeList = make([]ChangeLog, len(oldInfo.ChangeList))

copy(newInfo.ChangeList, oldInfo.ChangeList)

return nil

}

func (t \*SimpleChaincode) QuerySealLogoList(stub shim.ChaincodeStubInterface, args []string) pb.Response {

if len(args) != 1 {

msg := "{\"Result\": false, \"Msg\": \"Incorrect number of arguments. Expecting 1\"}"

return shim.Error(msg)

}

var input QuerySealLogoInputParam

var query string

//json str 转struct

if err := json.Unmarshal([]byte(args[0]), &input); err != nil {

\_, \_, line, \_ := runtime.Caller(0)

errMsg := fmt.Sprintf("Invalid input arg: %s. line: %d", args[0], line)

msg := fmt.Sprintf("{\"Result\": false, \"Msg\": \"%s\", \"SealLogoCount\":0, \"SealLogoInfoList\":[]}", errMsg)

return shim.Error(msg)

}

if input.SealType == 0 && input.SealLogoRecordCodeHash == "" { // 查询该企业的所有印模

query = fmt.Sprintf("{\"selector\":{\"EntCode\":\"%s\"}}", input.EntCode)

} else if input.SealType > 0 && input.SealLogoRecordCodeHash == "" {// 查询该企业的指定印章类型的印模

query = fmt.Sprintf("{\"selector\":{\"EntCode\":\"%s\", \"SealType\":%d}}", input.EntCode, input.SealType)

} else if input.SealType > 0 && input.SealLogoRecordCodeHash != "" { // 查询该企业的指定印章类型和指定印模备案编码的印模

query = fmt.Sprintf("{\"selector\":{\"EntCode\":\"%s\", \"SealType\":%d, \"SealLogoRecordCodeHash\":\"%s\"}}", input.EntCode, input.SealType, input.SealLogoRecordCodeHash)

} else {

\_, \_, line, \_ := runtime.Caller(0)

msg := fmt.Sprintf("{\"Result\": false, \"Msg\": \"Invalid input arg: %s. line: %d\", \"SealLogoCount\":0, \"SealLogoInfoList\":[]}", args[0], line)

return shim.Error(msg)

}

resultsIterator, err := stub.GetQueryResult(query)

if resultsIterator == nil {

\_, \_, line, \_ := runtime.Caller(0)

msg := fmt.Sprintf("{\"Result\": false, \"Msg\": \"找不到该印模. 企业统一社会信用代码:%s, 印章类型:%d, 印模备案编码:%s. line: %d\"}", input.EntCode, input.SealType, input.SealLogoRecordCodeHash, line)

return shim.Success([]byte(msg))

}

defer resultsIterator.Close()

if err != nil {

\_, \_, line, \_ := runtime.Caller(0)

msg := fmt.Sprintf("{\"Result\": false, \"Msg\": \"%s. line: %d\", \"SealLogoCount\":0, \"SealLogoInfoList\":[]}", err.Error(), line)

return shim.Error(msg)

}

var logoList []SealLogoInfo

var SealLogoCount = 0

for resultsIterator.HasNext() {

SealLogoCount++

logo := &SealLogoInfo{}

queryResponse, err := resultsIterator.Next()

if err != nil {

\_, \_, line, \_ := runtime.Caller(0)

msg := fmt.Sprintf("{\"Result\": false, \"Msg\": \"%s. line: %d\", \"SealLogoCount\":0, \"SealLogoInfoList\":[]}", err.Error(), line)

return shim.Error(msg)

}

//fmt.Println("key==>>", queryResponse.Key, "value===>>", queryResponse.Value)

json.Unmarshal(queryResponse.Value, &logo)

logoList = append(logoList, \*logo)

}

b, err5 := json.Marshal(logoList)

if err5 != nil {

//出错

\_, \_, line, \_ := runtime.Caller(0)

msg := fmt.Sprintf("{\"Result\": false, \"Msg\": \"%s. line: %d\", \"SealLogoCount\": %d, \"SealLogoInfoList\": []}", err5.Error(), line, SealLogoCount)

return shim.Success([]byte(msg))

}

if SealLogoCount == 0 {

msg := fmt.Sprintf("{\"Result\": true, \"Msg\": \"200\", \"SealLogoCount\": 0, \"SealLogoInfoList\": []}")

return shim.Success([]byte(msg))

} else {

msg := fmt.Sprintf("{\"Result\": true, \"Msg\": \"200\", \"SealLogoCount\": %d, \"SealLogoInfoList\": %s}", SealLogoCount, string(b))

return shim.Success([]byte(msg))

}

}

func (t \*SimpleChaincode) DisableSealLogo(stub shim.ChaincodeStubInterface, args []string) pb.Response {

if len(args) != 1 {

msg := "{\"Result\": false, \"Msg\": \"Incorrect number of arguments. Expecting 1\"}"

return shim.Error(msg)

}

var input QuerySealLogoInputParam

var newInfo SealLogoInfo

//json str 转struct

if err := json.Unmarshal([]byte(args[0]), &input); err != nil {

\_, \_, line, \_ := runtime.Caller(0)

msg := fmt.Sprintf("{\"Result\": false, \"Msg\": \"Invalid input arg: %s. line: %d\", \"SealLogoCount\":0, \"SealLogoInfoList\":[]}", args[0], line)

return shim.Error(msg)

}

if input.EntCode == "" {

\_, \_, line, \_ := runtime.Caller(0)

msg := fmt.Sprintf("10003：EntCode参数值需不能为空. line: %d", line)

return shim.Error(msg)

}

// 检查每个新印章的印章类型是否合法

if input.SealType < 1 || (input.SealType > 6 && input.SealType != 99) { //印章类型有：1 单位专用印章 2 财务专用章 3 税务专用章 4 合同专用章 5 法定代表人名章 6 名章 99 其他类型电子印章

\_, \_, line, \_ := runtime.Caller(0)

msg := fmt.Sprintf("40004：该印章类型不合法。印章类型：%d。 印章类型合法值范围 1-6和99. line: %d", input.SealType, line)

return shim.Error(msg)

}

if input.SealLogoRecordCodeHash == "" {

\_, \_, line, \_ := runtime.Caller(0)

msg := fmt.Sprintf("10003：SealLogoRecordCodeHash参数值需不能为空. line: %d", line)

return shim.Error(msg)

}

// 先检查记录是否已存在，如果是，添加更改记录

key := SealLogoKeyPrefix + newInfo.EntCode + "\_" + strconv.Itoa(int(newInfo.SealType)) + "\_" + newInfo.SealLogoRecordCodeHash

bytes, err3 := stub.GetState(key)

if err3 != nil {

\_, \_, line, \_ := runtime.Caller(0)

msg := fmt.Sprintf("{\"Result\": false, \"Msg\": \"%s\". line: %d\"}", err3.Error(), line)

return shim.Error(msg)

}

if bytes != nil { // 原来有记录

var oldInfo SealLogoInfo

if err := json.Unmarshal(bytes, &oldInfo); err != nil {

\_, \_, line, \_ := runtime.Caller(0)

msg := fmt.Sprintf("error：%s. line: %d", err.Error(), line)

return shim.Error(msg)

}

if oldInfo.SealLogoStatus == 0 {

\_, \_, line, \_ := runtime.Caller(0)

msg := fmt.Sprintf("{\"Result\": false, \"Msg\": \"该印模已是失效状态. line: %d\"}", line)

return shim.Error(msg)

}

// 印模更改记录处理

var operationType uint = 2 // 更新状态为失效

err := SealLogoChangeLogHandle(stub, key, operationType, &oldInfo, &newInfo)

if err != nil {

\_, \_, line, \_ := runtime.Caller(0)

msg := fmt.Sprintf("{\"Result\": false, \"Msg\": \"error: %s, line: %d\"}", err.Error(), line)

return shim.Error(msg)

}

} else { // 原来没有记录

\_, \_, line, \_ := runtime.Caller(0)

msg := fmt.Sprintf("{\"Result\": false, \"Msg\": \"找不到该印模. 企业统一社会信用代码:%s, 印章类型:%d, 印模备案编码:%s. line: %d\"}", input.EntCode, input.SealType, input.SealLogoRecordCodeHash, line)

return shim.Error(msg)

}

// 写db

newInfo.SealLogoStatus = 0

bytes, err2 := json.Marshal(newInfo);

if err2 != nil {

\_, \_, line, \_ := runtime.Caller(0)

msg := fmt.Sprintf("{\"Result\": false, \"Msg\": \"error: %s, line: %d\"}", err2.Error(), line)

return shim.Error(msg)

}

err := stub.PutState(key, bytes)

if err != nil {

\_, \_, line, \_ := runtime.Caller(0)

msg := fmt.Sprintf("{\"Result\": false, \"Msg\": \"error: %s, line: %d\"}", err.Error(), line)

return shim.Error(msg)

}

msg := fmt.Sprintf("{\"Result\": true, \"Msg\": \"200\"}")

return shim.Success([]byte(msg))

}

func (t \*SimpleChaincode) AddAccountantSealLogo(stub shim.ChaincodeStubInterface, args []string) pb.Response {

if len(args) != 1 {

Msg := "{\"Result\": false, \"Msg\": \"Incorrect number of arguments. Expecting 1\", \"OrderId\",\"\"}"

return shim.Error(Msg)

}

var newInfo PersonalSealLogoInfo

//json str 转struct

if err := json.Unmarshal([]byte(args[0]), &newInfo); err != nil {

\_, \_, line, \_ := runtime.Caller(0)

errMsg := fmt.Sprintf("Invalid input arg: %s", args[0])

msg := fmt.Sprintf("{\"Result\": false, \"Msg\": \"%s. line: %d\"}", errMsg, line)

return shim.Error(msg)

}

// 新增个人的印章印模预处理

if err := AddPersonalSealLogoPreHandle(stub, &newInfo); err != nil {

msg := fmt.Sprintf("{\"Result\": false, \"Msg\": \"%s\"}", err.Error())

return shim.Error(msg)

}

// 先检查记录是否已存在，如果是，添加更改记录

key := PersonalSealLogoKeyPrefix + strconv.Itoa(int(newInfo.IdentityType)) + "\_" + newInfo.IdentityNo + "\_" + strconv.Itoa(int(newInfo.SealType))

bytes, err3 := stub.GetState(key)

if err3 != nil {

\_, \_, line, \_ := runtime.Caller(0)

msg := fmt.Sprintf("{\"Result\": false, \"Msg\": \"%s\". line: %d\"}", err3.Error(), line)

return shim.Error(msg)

}

if bytes != nil { // 原来有记录

var oldInfo PersonalSealLogoInfo

if err := json.Unmarshal(bytes, &oldInfo); err != nil {

\_, \_, line, \_ := runtime.Caller(0)

msg := fmt.Sprintf("error：%s. line: %d", err.Error(), line)

return shim.Error(msg)

}

// 印模更改记录处理

var operationType uint = 1 //更新文件

err := PersonalSealLogoChangeLogHandle(stub, key, operationType, &oldInfo, &newInfo)

if err != nil {

\_, \_, line, \_ := runtime.Caller(0)

msg := fmt.Sprintf("{\"Result\": false, \"Msg\": \"error: %s, line: %d\"}", err.Error(), line)

return shim.Error(msg)

}

}

bytes2, err2 := json.Marshal(newInfo);

if err2 != nil {

\_, \_, line, \_ := runtime.Caller(0)

msg := fmt.Sprintf("{\"Result\": false, \"Msg\": \"error: %s, line: %d\"}", err2.Error(), line)

return shim.Error(msg)

}

// 写db

err := stub.PutState(key, bytes2)

if err != nil {

\_, \_, line, \_ := runtime.Caller(0)

msg := fmt.Sprintf("{\"Result\": false, \"Msg\": \"error: %s, line: %d\"}", err.Error(), line)

return shim.Error(msg)

}

msg := fmt.Sprintf("{\"Result\": true, \"Msg\": \"200\"}")

return shim.Success([]byte(msg))

}

func AddPersonalSealLogoPreHandle(stub shim.ChaincodeStubInterface, info \*PersonalSealLogoInfo) error {

//if info.IdentityType < 1 || info.IdentityType > 3 {

if info.IdentityType != 1 {

\_, \_, line, \_ := runtime.Caller(0)

msg := fmt.Sprintf("10003：IdentityType参数值应为1. line: %d", line)

return errors.New(msg)

}

if info.IdentityNo == "" {

\_, \_, line, \_ := runtime.Caller(0)

msg := fmt.Sprintf("10003：IdentityNo参数值需不能为空. line: %d", line)

return errors.New(msg)

}

// 检查每个新印章的印章类型是否合法

if info.SealType != 2001 {// 注册会计师资格章

\_, \_, line, \_ := runtime.Caller(0)

msg := fmt.Sprintf("40004：该印章类型不合法。印章类型：%d。 印章类型合法值是 2001. line: %d", info.SealType, line)

return errors.New(msg)

}

if info.SealLogoFileUrl == "" {

\_, \_, line, \_ := runtime.Caller(0)

msg := fmt.Sprintf("10003：SealLogoFileUrl参数值需不能为空. line: %d", line)

return errors.New(msg)

}

if info.SealLogoFileHash == "" {

\_, \_, line, \_ := runtime.Caller(0)

msg := fmt.Sprintf("10003：SealLogoFileHash参数值需不能为空. line: %d", line)

return errors.New(msg)

}

/\* if info.SealLogoExpireDate <= 0 {

\_, \_, line, \_ := runtime.Caller(0)

msg := fmt.Sprintf("10003：SealLogoExpireDate参数值需大于0. line: %d", line)

return errors.New(msg)

}

\*/

if info.AppId == "" {

\_, \_, line, \_ := runtime.Caller(0)

msg := fmt.Sprintf("10003：AppId参数值需不能为空. line: %d", line)

return errors.New(msg)

}

info.SealLogoStatus = 1 // 印模状态 0 失效 1 有效

return nil

}

func PersonalSealLogoChangeLogHandle(stub shim.ChaincodeStubInterface, key string, operationType uint, oldInfo, newInfo \*PersonalSealLogoInfo) error {

// 添加更改终端用户状态记录

t1, err4 := stub.GetTxTimestamp()

if err4 != nil {

\_, \_, line, \_ := runtime.Caller(0)

msg := fmt.Sprintf("{\"Result\": false, \"Msg\": \"GetTxTimestamp error:%s, line: %d\"}", err4.Error(), line)

return errors.New(msg)

}

var log ChangeLog

switch(operationType){

case 1:

log.Operation = "更新文件"

break;

case 2:

log.Operation = "更新状态为失效"

\*newInfo = \*oldInfo

break

default:

\_, \_, line, \_ := runtime.Caller(0)

msg := fmt.Sprintf("{\"Result\": false, \"Msg\": \"error: wrong operationType%s, line: %d\"}",operationType, line)

return errors.New(msg)

}

log.OperTime = time.Unix(t1.Seconds, int64(t1.Nanos)).UnixNano() / 1e6 // unix时间戳，精确到毫秒。

log.TxId = stub.GetTxID()

log.OldFileUrl = oldInfo.SealLogoFileUrl

log.OldFileHash = oldInfo.SealLogoFileHash

oldInfo.ChangeList = append(oldInfo.ChangeList, log)

newInfo.ChangeList = make([]ChangeLog, len(oldInfo.ChangeList))

copy(newInfo.ChangeList, oldInfo.ChangeList)

return nil

}

func (t \*SimpleChaincode) QueryPersonalSealLogoList(stub shim.ChaincodeStubInterface, args []string) pb.Response {

if len(args) != 1 {

msg := "{\"Result\": false, \"Msg\": \"Incorrect number of arguments. Expecting 1\"}"

return shim.Error(msg)

}

var input QueryPersonalSealLogoInputParam

var query string

//json str 转struct

if err := json.Unmarshal([]byte(args[0]), &input); err != nil {

\_, \_, line, \_ := runtime.Caller(0)

errMsg := fmt.Sprintf("Invalid input arg: %s. line: %d", args[0], line)

msg := fmt.Sprintf("{\"Result\": false, \"Msg\": \"%s\", \"SealLogoCount\":0, \"SealLogoInfoList\":[]}", errMsg)

return shim.Error(msg)

}

if input.SealType == 0 { // 查询该企业的所有印模

query = fmt.Sprintf("{\"selector\":{\"IdentityType\":%d, \"IdentityNo\":\"%s\", \"SealLogoStatus\":1}}", input.IdentityType, input.IdentityNo)

} else if input.SealType > 0 {// 查询该企业的指定印章类型的印模

query = fmt.Sprintf("{\"selector\":{\"IdentityType\":%d, \"IdentityNo\":\"%s\", \"SealType\":%d, \"SealLogoStatus\":1}}", input.IdentityType, input.IdentityNo, input.SealType)

} else {

\_, \_, line, \_ := runtime.Caller(0)

msg := fmt.Sprintf("{\"Result\": false, \"Msg\": \"Invalid input arg: %s. line: %d\", \"SealLogoCount\":0, \"SealLogoInfoList\":[]}", args[0], line)

return shim.Error(msg)

}

resultsIterator, err := stub.GetQueryResult(query)

if resultsIterator == nil {

\_, \_, line, \_ := runtime.Caller(0)

msg := fmt.Sprintf("{\"Result\": false, \"Msg\": \"找不到该印模. 证件类型:%d, 证件号:%s, 印章类型:%d. line: %d\"}", input.IdentityType, input.IdentityNo, input.SealType, line)

return shim.Success([]byte(msg))

}

defer resultsIterator.Close()

if err != nil {

\_, \_, line, \_ := runtime.Caller(0)

msg := fmt.Sprintf("{\"Result\": false, \"Msg\": \"%s. line: %d\", \"SealLogoCount\":0, \"SealLogoInfoList\":[]}", err.Error(), line)

return shim.Error(msg)

}

var logoList []SealLogoInfo

var SealLogoCount = 0

for resultsIterator.HasNext() {

SealLogoCount++

logo := &SealLogoInfo{}

queryResponse, err := resultsIterator.Next()

if err != nil {

\_, \_, line, \_ := runtime.Caller(0)

msg := fmt.Sprintf("{\"Result\": false, \"Msg\": \"%s. line: %d\", \"SealLogoCount\":0, \"SealLogoInfoList\":[]}", err.Error(), line)

return shim.Error(msg)

}

//fmt.Println("key==>>", queryResponse.Key, "value===>>", queryResponse.Value)

json.Unmarshal(queryResponse.Value, &logo)

logoList = append(logoList, \*logo)

}

b, err5 := json.Marshal(logoList)

if err5 != nil {

//出错

\_, \_, line, \_ := runtime.Caller(0)

msg := fmt.Sprintf("{\"Result\": false, \"Msg\": \"%s. line: %d\", \"SealLogoCount\": %d, \"SealLogoInfoList\": []}", err5.Error(), line, SealLogoCount)

return shim.Success([]byte(msg))

}

if SealLogoCount == 0 {

msg := fmt.Sprintf("{\"Result\": true, \"Msg\": \"200\", \"SealLogoCount\": 0, \"SealLogoInfoList\": []}")

return shim.Success([]byte(msg))

} else {

msg := fmt.Sprintf("{\"Result\": true, \"Msg\": \"200\", \"SealLogoCount\": %d, \"SealLogoInfoList\": %s}", SealLogoCount, string(b))

return shim.Success([]byte(msg))

}

}