import com.mathworks.toolbox.javabuilder.Disposable;

import com.mathworks.toolbox.javabuilder.MWComponentOptions;

import com.mathworks.toolbox.javabuilder.MWCtfDirectorySource;

import com.mathworks.toolbox.javabuilder.MWCtfExtractLocation;

import com.mathworks.toolbox.javabuilder.MWException;

import com.mathworks.toolbox.javabuilder.internal.MWComponentInstance;

import com.mathworks.toolbox.javabuilder.internal.MWFunctionSignature;

import com.mathworks.toolbox.javabuilder.internal.MWMCR;

import java.io.Serializable;

import java.util.Arrays;

import java.util.HashSet;

import java.util.Iterator;

import java.util.List;

import java.util.Set;

public class Harmonic\_stability extends MWComponentInstance<Harmonic\_stability> implements Serializable {

private static final Set<Disposable> sInstances = new HashSet();

private static final MWFunctionSignature sA\_dqimpedanceSignature = new MWFunctionSignature(0, false, "A\_dqimpedance", 0, false);

private static final MWFunctionSignature sA\_notationSignature = new MWFunctionSignature(0, false, "A\_notation", 0, false);

private static final MWFunctionSignature sA\_pnimpedanceSignature = new MWFunctionSignature(0, false, "A\_pnimpedance", 0, false);

private static final MWFunctionSignature sA\_SISOpnimpedanceSignature = new MWFunctionSignature(0, false, "A\_SISOpnimpedance", 0, false);

private static final MWFunctionSignature sB\_dqimpedanceSignature = new MWFunctionSignature(0, false, "B\_dqimpedance", 0, false);

private static final MWFunctionSignature sB\_notationSignature = new MWFunctionSignature(0, false, "B\_notation", 0, false);

private static final MWFunctionSignature sB\_pnimpedanceSignature = new MWFunctionSignature(0, false, "B\_pnimpedance", 0, false);

private static final MWFunctionSignature sB\_SISOpnimpedanceSignature = new MWFunctionSignature(0, false, "B\_SISOpnimpedance", 0, false);

private Harmonic\_stability(MWMCR var1) throws MWException {

super(var1);

Class var2 = Harmonic\_stability.class;

synchronized(Harmonic\_stability.class) {

sInstances.add(this);

}

}

public Harmonic\_stability() throws MWException {

this(STRUCTUREMCRFactory.newInstance());

}

private static MWComponentOptions getPathToComponentOptions(String var0) {

MWComponentOptions var1 = new MWComponentOptions(new Object[]{new MWCtfExtractLocation(var0), new MWCtfDirectorySource(var0)});

return var1;

}

/\*\* @deprecated \*/

@Deprecated

public Harmonic\_stability(String var1) throws MWException { this(STRUCTUREMCRFactory.newInstance(getPathToComponentOptions(var1)));

}

public Harmonic\_stability(MWComponentOptions var1) throws MWException {

this(STRUCTUREMCRFactory.newInstance(var1));

}

public void dispose() {

boolean var9 = false;

try {

var9 = true;

super.dispose();

var9 = false;

} finally {

if (var9) {

Class var4 = Harmonic\_stability.class;

synchronized(Harmonic\_stability.class) {

sInstances.remove(this);

}

}

}

Class var1 = Harmonic\_stability.class;

synchronized(Harmonic\_stability.class) {

sInstances.remove(this);

}

}

public static void disposeAllInstances() {

Class var0 = Harmonic\_stability.class;

synchronized(Harmonic\_stability.class) {

Iterator var1 = sInstances.iterator();

while(var1.hasNext()) {

Disposable var2 = (Disposable)var1.next();

var2.dispose();

}

sInstances.clear();

}

}

public void A\_dqimpedance(List var1, List var2) throws MWException {

this.fMCR.invoke(var1, var2, sA\_dqimpedanceSignature);

}

public void A\_dqimpedance(Object[] var1, Object[] var2) throws MWException {

this.fMCR.invoke(Arrays.asList(var1), Arrays.asList(var2), sA\_dqimpedanceSignature);

}

public Object[] A\_dqimpedance(Object... var1) throws MWException {

Object[] var2 = new Object[0];

this.fMCR.invoke(Arrays.asList(var2), MWMCR.getRhsCompat(var1, sA\_dqimpedanceSignature), sA\_dqimpedanceSignature);

return var2;

}

public void A\_notation(List var1, List var2) throws MWException {

this.fMCR.invoke(var1, var2, sA\_notationSignature);

}

public void A\_notation(Object[] var1, Object[] var2) throws MWException {

this.fMCR.invoke(Arrays.asList(var1), Arrays.asList(var2), sA\_notationSignature);

}

public Object[] A\_notation(Object... var1) throws MWException {

Object[] var2 = new Object[0];

this.fMCR.invoke(Arrays.asList(var2), MWMCR.getRhsCompat(var1, sA\_notationSignature), sA\_notationSignature);

return var2;

}

public void A\_pnimpedance(List var1, List var2) throws MWException {

this.fMCR.invoke(var1, var2, sA\_pnimpedanceSignature);

}

public void A\_pnimpedance(Object[] var1, Object[] var2) throws MWException {

this.fMCR.invoke(Arrays.asList(var1), Arrays.asList(var2), sA\_pnimpedanceSignature);

}

public Object[] A\_pnimpedance(Object... var1) throws MWException {

Object[] var2 = new Object[0];

this.fMCR.invoke(Arrays.asList(var2), MWMCR.getRhsCompat(var1, sA\_pnimpedanceSignature), sA\_pnimpedanceSignature);

return var2;

}

public void A\_SISOpnimpedance(List var1, List var2) throws MWException {

this.fMCR.invoke(var1, var2, sA\_SISOpnimpedanceSignature);

}

public void A\_SISOpnimpedance(Object[] var1, Object[] var2) throws MWException {

this.fMCR.invoke(Arrays.asList(var1), Arrays.asList(var2), sA\_SISOpnimpedanceSignature);

}

public Object[] A\_SISOpnimpedance(Object... var1) throws MWException {

Object[] var2 = new Object[0];

this.fMCR.invoke(Arrays.asList(var2), MWMCR.getRhsCompat(var1, sA\_SISOpnimpedanceSignature), sA\_SISOpnimpedanceSignature);

return var2;

}

public void B\_dqimpedance(List var1, List var2) throws MWException {

this.fMCR.invoke(var1, var2, sB\_dqimpedanceSignature);

}

public void B\_dqimpedance(Object[] var1, Object[] var2) throws MWException {

this.fMCR.invoke(Arrays.asList(var1), Arrays.asList(var2), sB\_dqimpedanceSignature);

}

public Object[] B\_dqimpedance(Object... var1) throws MWException {

Object[] var2 = new Object[0];

this.fMCR.invoke(Arrays.asList(var2), MWMCR.getRhsCompat(var1, sB\_dqimpedanceSignature), sB\_dqimpedanceSignature);

return var2;

}

public void B\_notation(List var1, List var2) throws MWException {

this.fMCR.invoke(var1, var2, sB\_notationSignature);

}

public void B\_notation(Object[] var1, Object[] var2) throws MWException {

this.fMCR.invoke(Arrays.asList(var1), Arrays.asList(var2), sB\_notationSignature);

}

public Object[] B\_notation(Object... var1) throws MWException {

Object[] var2 = new Object[0];

this.fMCR.invoke(Arrays.asList(var2), MWMCR.getRhsCompat(var1, sB\_notationSignature), sB\_notationSignature);

return var2;

}

public void B\_pnimpedance(List var1, List var2) throws MWException {

this.fMCR.invoke(var1, var2, sB\_pnimpedanceSignature);

}

public void B\_pnimpedance(Object[] var1, Object[] var2) throws MWException {

this.fMCR.invoke(Arrays.asList(var1), Arrays.asList(var2), sB\_pnimpedanceSignature);

}

public Object[] B\_pnimpedance(Object... var1) throws MWException {

Object[] var2 = new Object[0];

this.fMCR.invoke(Arrays.asList(var2), MWMCR.getRhsCompat(var1, sB\_pnimpedanceSignature), sB\_pnimpedanceSignature);

return var2;

}

public void B\_SISOpnimpedance(List var1, List var2) throws MWException {

this.fMCR.invoke(var1, var2, sB\_SISOpnimpedanceSignature);

}

public void B\_SISOpnimpedance(Object[] var1, Object[] var2) throws MWException {

this.fMCR.invoke(Arrays.asList(var1), Arrays.asList(var2), sB\_SISOpnimpedanceSignature);

}

public Object[] B\_SISOpnimpedance(Object... var1) throws MWException {

Object[] var2 = new Object[0];

this.fMCR.invoke(Arrays.asList(var2), MWMCR.getRhsCompat(var1, sB\_SISOpnimpedanceSignature), sB\_SISOpnimpedanceSignature);

return var2;

}

}

import com.mathworks.toolbox.javabuilder.MWComponentOptions;

import com.mathworks.toolbox.javabuilder.MWCtfClassLoaderSource;

import com.mathworks.toolbox.javabuilder.MWCtfExtractLocation;

import com.mathworks.toolbox.javabuilder.MWException;

import com.mathworks.toolbox.javabuilder.internal.MWMCR;

import java.io.Serializable;

public class STRUCTUREMCRFactory implements Serializable {

private static final String sComponentId = "STRUCTURE\_1307dd6b-03ac-45f9-942e-b3beefb1bfea";

private static final String sComponentName = "STRUCTURE";

private static final MWComponentOptions sDefaultComponentOptions;

private STRUCTUREMCRFactory() {

}

public static MWMCR newInstance(MWComponentOptions var0) throws MWException {

if (null == var0.getCtfSource()) {

var0 = new MWComponentOptions(var0);

var0.setCtfSource(sDefaultComponentOptions.getCtfSource());

}

return MWMCR.newInstance(var0, STRUCTUREMCRFactory.class, "STRUCTURE", "STRUCTURE\_1307dd6b-03ac-45f9-942e-b3beefb1bfea", new int[]{9, 12, 0});

}

public static MWMCR newInstance() throws MWException {

return newInstance(sDefaultComponentOptions);

}

static {

sDefaultComponentOptions = new MWComponentOptions(new Object[]{MWCtfExtractLocation.EXTRACT\_TO\_CACHE, new MWCtfClassLoaderSource(STRUCTUREMCRFactory.class)});

}

}

import com.ixunqi.dev.model.vo.\*;

import lombok.extern.slf4j.Slf4j;

import org.springframework.http.HttpStatus;

import org.springframework.http.ResponseEntity;

import org.springframework.http.converter.HttpMessageNotReadableException;

import org.springframework.validation.BindException;

import org.springframework.web.HttpMediaTypeNotSupportedException;

import org.springframework.web.HttpRequestMethodNotSupportedException;

import org.springframework.web.bind.MethodArgumentNotValidException;

import org.springframework.web.bind.MissingServletRequestParameterException;

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

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

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

/\*\*

\* Created on 2018/3/27 15:05

\* Description: []

\* @author

\*/

@Slf4j

@ControllerAdvice

public class GlobalExceptionHandler {

@ExceptionHandler

@ResponseBody

public ResponseEntity<Result<String>> validationExceptionHandle(BindException exception) {

StringBuilder msg = new StringBuilder();

exception.getAllErrors().forEach(objectError -> {

log.warn("参数异常！{}", objectError);

msg.append(objectError.getDefaultMessage()).append(",");

});

return ResponseEntity.status(HttpStatus.BAD\_REQUEST).body(Result.error(msg.toString()));

}

@ExceptionHandler(MethodArgumentNotValidException.class)

@ResponseBody

public ResponseEntity<Result<String>> validationExceptionHandle(MethodArgumentNotValidException exception) {

StringBuilder msg = new StringBuilder();

exception.getBindingResult().getAllErrors().forEach(objectError -> {

log.warn("参数异常！{}", objectError);

msg.append(objectError.getDefaultMessage()).append(",");

});

return ResponseEntity.status(HttpStatus.BAD\_REQUEST).body(Result.error(msg.toString()));

}

@ExceptionHandler(HttpRequestMethodNotSupportedException.class)

@ResponseBody

public ResponseEntity<Result<String>> exceptionHandle(HttpRequestMethodNotSupportedException e) {

log.info("http请求不支持该方法", e);

return ResponseEntity.status(HttpStatus.BAD\_REQUEST).body(Result.error("http请求不支持该方法"));

}

@ExceptionHandler(HttpMediaTypeNotSupportedException.class)

@ResponseBody

public ResponseEntity<Result<String>> exceptionHandle(HttpMediaTypeNotSupportedException e) {

log.info("Content type 错误", e);

return ResponseEntity.status(HttpStatus.BAD\_REQUEST).body(Result.error("content type 不支持！" + e.getContentType()));

}

@ExceptionHandler(MissingServletRequestParameterException.class)

@ResponseBody

public ResponseEntity<Result<String>> exceptionHandle(MissingServletRequestParameterException e) {

log.info("缺少必要参数", e);

return ResponseEntity.status(HttpStatus.BAD\_REQUEST).body(Result.error(e.getParameterName() + "不能为空"));

}

/\*\*

\* 请求体为空或者请求体json格式化错误会抛此异常

\*/

@ExceptionHandler(HttpMessageNotReadableException.class)

@ResponseBody

public ResponseEntity<Result<String>> exceptionHandle(HttpMessageNotReadableException e) {

log.warn("HttpMessageNotReadableException", e);

return ResponseEntity.status(HttpStatus.BAD\_REQUEST).body(Result.error("请求体json格式化错误"));

}

@ExceptionHandler(Exception.class)

@ResponseBody

public ResponseEntity<Result<String>> unknownExceptionHandle(Exception exception) {

log.error("未知异常！", exception);

return ResponseEntity.status(HttpStatus.INTERNAL\_SERVER\_ERROR).body(Result.error(exception.getMessage()));

}

}

import cn.hutool.core.date.DateField;

import cn.hutool.core.date.DateUtil;

import com.cet.pq.anlysis.model.qualtityevaluation.Line;

import com.cet.pq.common.model.Result;

import com.cet.pq.harmonic.trend.datachange.TrendResultDTOChange;

import com.cet.pq.harmonic.trend.datachange.WholeNetRelationChange;

import com.cet.pq.harmonic.trend.datachange.XddaonaDTOChange;

import com.cet.pq.harmonic.trend.mapper.TrendResultMapper;

import com.cet.pq.harmonic.trend.mapper.WholeNetMapMapper;

import com.cet.pq.harmonic.trend.mapper.WholeNetRelationMapper;

import com.cet.pq.harmonic.trend.model.\*;

import com.cet.pq.harmonic.trend.model.dto.JddaonaDTO;

import com.cet.pq.harmonic.trend.model.dto.TrendMapDataDTO;

import com.cet.pq.harmonic.trend.model.dto.TrendResultDTO;

import com.cet.pq.harmonic.trend.model.dto.XddaonaDTO;

import com.cet.pq.harmonic.trend.model.query.OverviewMapPsrQuery;

import com.cet.pq.harmonic.trend.model.query.OverviewMapQuery;

import com.cet.pq.harmonic.trend.service.CalculateService;

import com.cet.pq.harmonic.trend.service.NwSubstationService;

import com.cet.pq.harmonic.trend.service.WholeNetService;

import com.cet.pq.pqservice.service.LineService;

import com.xiaoleilu.hutool.bean.BeanUtil;

import org.apache.commons.collections4.CollectionUtils;

import org.apache.commons.lang3.StringUtils;

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

import org.springframework.stereotype.Service;

import org.springframework.transaction.annotation.Transactional;

import java.util.\*;

import java.util.stream.Collectors;

import java.util.stream.Stream;

import static java.util.Comparator.comparing;

import static java.util.stream.Collectors.collectingAndThen;

import static java.util.stream.Collectors.toCollection;

@Service

public class WholeNetServiceImpl implements WholeNetService {

@Autowired

WholeNetMapMapper wholeNetMapMapper;

@Autowired

WholeNetRelationMapper wholeNetRelationMapper;

@Autowired

TrendResultMapper trendResultMapper;

@Autowired

private LineService lineService;

@Autowired

private CalculateService calculateService;

@Autowired

private NwSubstationService nwSubstationService;

@Override

@Transactional(rollbackFor = Exception.class, transactionManager = "matterhornTransactionManager")

public Result init(OverviewMapPsrQuery query) {

Integer level = query.getLevel();

Integer code = query.getCode();

List<String> vlevels = query.getVlevels();

WholeNetMap netMap = wholeNetMapMapper.getOne(level, code);

List<WholeNetRelation> relationList = new ArrayList<>();

if (netMap == null) {

netMap = new WholeNetMap(level, code);

List<PsrDataDTO> psrDataDTOList = query.getPsrDataDTOList();

List<PsrDTO> substationPsrDTOList = getSubstationPsrDTOList(psrDataDTOList, vlevels);

if (CollectionUtils.isEmpty(substationPsrDTOList)) {

return new Result(NO\_FEEDER\_CODE, NO\_FEEDER\_DESC, null);

}

List<PsrDTO> linePsrDTOList = getLinePsrDTOList(substationPsrDTOList);

relationList = getWholeNetRelations(linePsrDTOList, netMap.getId());

if (!saveMapAndRelation(netMap, relationList)) {

return new Result(NO\_FEEDER\_CODE, NO\_FEEDER\_DESC, null);

}

calculate(query, relationList);

}

return Result.success();

}

@Override

public Result<TrendMapDataDTO> queryOrCreateTrendResult(OverviewMapQuery query) {

Integer trendTime = query.getTrendTime();

Date trendDate = query.getTrendDate();

trendDate = DateUtil.offsetHour(trendDate, trendTime);

query.setTrendDate(trendDate);

WholeNetMap wholeNetMap = this.getOneMap(query.getLevel(), query.getCode());

if (wholeNetMap == null) {

return new Result(NO\_MAP\_CODE, NO\_MAP\_DESC, null);

}

List<WholeNetRelation> wholeNetRelations = this.relationListByMapId(wholeNetMap.getId());

if (CollectionUtils.isEmpty(wholeNetRelations)) {

return new Result(NO\_MAP\_CODE, NO\_MAP\_DESC, null);

}

TrendMapDataDTO trendMapDataDTO = this.queryTrendResult(query);

if (trendMapDataDTO == null) {

synchronized (this) {

trendMapDataDTO = this.queryTrendResult(query);

if (trendMapDataDTO == null) {

this.calculate(query, wholeNetRelations);

trendMapDataDTO = this.queryTrendResult(query);

}

}

}

return Result.success(trendMapDataDTO);

}

public TrendMapDataDTO queryTrendResult(OverviewMapQuery query) {

List<TrendResultDTO> trendResultDTOS = trendResultMapper.wholeNetTrendResult(query);

return TrendResultDTOChange.toTrendMapDataDTO(trendResultDTOS);

}

@Override

public void calculate(OverviewMapQuery query, List<WholeNetRelation> relationList) {

// WholeNetRelation 转换为 XddaonaDTO

List<XddaonaDTO> xddaonaDTOS = WholeNetRelationChange.toXddaonaDTO(relationList);

List<JddaonaDTO> jddaonaDTOS = XddaonaDTOChange.toJddaonaDTO(xddaonaDTOS);

List<Long> nodeIdList = jddaonaDTOS.stream().map(JddaonaDTO::getNodeId).collect(Collectors.toList());

List<Line> lines = lineService.selectListByNodeIds(nodeIdList);

Integer harminocNum = query.getTrendCount();

Date trendDate = query.getTrendDate();

Integer trendTime = query.getTrendTime();

Date startDateTime = DateUtil.beginOfDay(trendDate);

Date endDateTime = null;

if (trendTime == null) {

endDateTime = DateUtil.endOfDay(trendDate);

} else {

startDateTime = DateUtil.offsetHour(startDateTime, trendTime);

endDateTime = DateUtil.offsetHour(startDateTime, 1).offset(DateField.SECOND, -1);

}

calculateService.calculate(xddaonaDTOS, jddaonaDTOS, lines, harminocNum, startDateTime, endDateTime);

}

@Override

public boolean saveMapAndRelation(WholeNetMap netMap, List<WholeNetRelation> relationList) {

if (netMap == null || CollectionUtils.isEmpty(relationList)) {

return false;

}

this.saveRelationList(relationList, netMap.getId());

this.saveMap(netMap);

return true;

}

@Override

public WholeNetMap saveMap(WholeNetMap netMap) {

WholeNetMap localData = wholeNetMapMapper.getOne(netMap.getLevel(), netMap.getCode());

if (localData != null) {

return localData;

}

synchronized (this) {

localData = wholeNetMapMapper.getOne(netMap.getLevel(), netMap.getCode());

if (localData != null) {

return localData;

}

wholeNetMapMapper.save(netMap);

}

return netMap;

}

/\*\*

\* 获取图形结构

\* @param relationList

\* @return

\*/

@Override

public List<WholeNetRelation> saveRelationList(List<WholeNetRelation> relationList, Long mapId) {

if (CollectionUtils.isEmpty(relationList)) {

return null;

}

synchronized (this) {

List<WholeNetRelation> wholeNetRelations = wholeNetRelationMapper.listByMapId(mapId);

if (CollectionUtils.isNotEmpty(wholeNetRelations)) {

return wholeNetRelations;

}

wholeNetRelationMapper.saveBatch(relationList);

}

return relationList;

}

/\*\*

\* @param linePsrDTOList

\* @return

\*//\*

@Override

public List<WholeNetRelation> saveRelationList(List<PsrDTO> linePsrDTOList, Long mapId) {

List<WholeNetRelation> relationList = getWholeNetRelations(linePsrDTOList, mapId);

if (CollectionUtils.isEmpty(relationList)) {

return null;

}

synchronized (this) {

List<WholeNetRelation> wholeNetRelations = wholeNetRelationMapper.listByMapId(mapId);

if (CollectionUtils.isNotEmpty(wholeNetRelations)) {

return wholeNetRelations;

}

wholeNetRelationMapper.saveBatch(relationList);

}

return relationList;

}\*/

/\*\*

\* @param linePsrDTOList

\* @param mapId

\* @return

\*/

private List<WholeNetRelation> getWholeNetRelations(List<PsrDTO> linePsrDTOList, Long mapId) {

if (CollectionUtils.isEmpty(linePsrDTOList)) {

return null;

}

Set<String> startDevIds = linePsrDTOList.stream().map(PsrDTO::getStartDevId).collect(Collectors.toSet());

Set<String> endDevIds = linePsrDTOList.stream().map(PsrDTO::getEndDevId).collect(Collectors.toSet());

Set<String> devIds = Stream.concat(startDevIds.stream(), endDevIds.stream()).collect(Collectors.toSet());

List<Substation> substations = nwSubstationService.listByDevIds(devIds);

if (CollectionUtils.isEmpty(substations)) {

return null;

}

List<WholeNetRelation> relationList= new ArrayList<>();

for (PsrDTO line : linePsrDTOList) {

String startDevId = line.getStartDevId();

String endDevId = line.getEndDevId();

Substation startSubstation = substations.stream().filter(substation -> StringUtils.equals(startDevId, substation.getDevId())).findAny().orElse(null);

Substation endSubstation = substations.stream().filter(substation -> StringUtils.equals(endDevId, substation.getDevId())).findAny().orElse(null);

if (startSubstation == null || endSubstation == null) {

continue;

}

WholeNetRelation relation = new WholeNetRelation();

BeanUtil.copyProperties(line, relation);

relation.setMapId(mapId);

relation.setStartId(startSubstation.getId());

relation.setEndId(endSubstation.getId());

Double length = relation.getLength();

if (length != null) {

relation.setLength(length\*1000D);

}

relationList.add(relation);

}

if (CollectionUtils.isEmpty(relationList)) {

return relationList;

}

relationList = relationList.stream().filter(item -> StringUtils.isNotEmpty(item.getDevId())).collect(

collectingAndThen(

toCollection(() -> new TreeSet<>(comparing(WholeNetRelation::getDevId))), ArrayList::new)

);

return relationList;

}

@Override

public WholeNetMap getOneMap(Integer level, Integer code) {

return wholeNetMapMapper.getOne(level, code);

}

@Override

public List<WholeNetRelation> relationListByMapId(Long mapId) {

return wholeNetRelationMapper.listByMapId(mapId);

}

/\*\*

\* @param substationPsrDTOList

\* @return

\*/

private static List<PsrDTO> getLinePsrDTOList(List<PsrDTO> substationPsrDTOList) {

if (CollectionUtils.isEmpty(substationPsrDTOList)) {

return null;

}

Set<String> substationDevIdSet = substationPsrDTOList.stream().map(PsrDTO::getDevId).collect(Collectors.toSet());

List<PsrDTO> linePsrDTOList = new ArrayList<>();

for (PsrDTO psrDTO : substationPsrDTOList) {

List<PsrDataDTO> psrDataLineList = psrDTO.getPsrDataDTOList();

if (CollectionUtils.isEmpty(psrDataLineList)) {

continue;

}

for (PsrDataDTO psrDataLine : psrDataLineList) {

List<PsrDTO> psrLineList = psrDataLine.getPsrList();

if (CollectionUtils.isEmpty(psrLineList)) {

continue;

}

for (PsrDTO line : psrLineList) {

String startDevId = line.getStartDevId();

String endDevId = line.getEndDevId();

if (StringUtils.isEmpty(startDevId) || StringUtils.isEmpty(endDevId)) {

continue;

}

if (!substationDevIdSet.contains(startDevId) || !substationDevIdSet.contains(endDevId)) {

continue;

}

if (!linePsrDTOList.contains(line)) {

linePsrDTOList.add(line);

}

}

}

}

return linePsrDTOList;

}

/\*\*

\* @param psrDataDTOList

\* @param vlevels 电压等级 过滤条件

\* @return

\*/

private List<PsrDTO> getSubstationPsrDTOList(List<PsrDataDTO> psrDataDTOList, List<String> vlevels) {

List<PsrDTO> substationPsrDTOList = new ArrayList<>();

for (PsrDataDTO psrDataDTO : psrDataDTOList) {

List<PsrDTO> substationList = psrDataDTO.getPsrList();

if (CollectionUtils.isEmpty(substationList)) {

continue;

}

substationPsrDTOList.addAll(substationList);

}

if (CollectionUtils.isEmpty(substationPsrDTOList)) {

return null;

}

substationPsrDTOList = substationPsrDTOList.stream()

.filter(item -> StringUtils.isNotEmpty(item.getDevId()))

.filter(item -> {

if (CollectionUtils.isEmpty(vlevels)) {

return true;

}

return vlevels.contains(item.getVlevelName());

})

.collect(

collectingAndThen(

toCollection(() -> new TreeSet<>(comparing(PsrDTO::getDevId))), ArrayList::new)

);

return substationPsrDTOList;

}

}

import com.cet.pq.common.utils.uuid.IdUtils;

import com.cet.pq.harmonicAnlysis.entity.DTO.NwHarmonicDataDTO;

import com.cet.pq.harmonicAnlysis.entity.LineInfo;

import com.cet.pq.harmonicAnlysis.entity.SubstationTypeEnum;

import com.cet.pq.harmonicAnlysis.entity.VO.NwHarmonicDataVO;

import com.cet.pq.harmonicAnlysis.entity.VO.NwHarmonicProvince;

import com.cet.pq.harmonicAnlysis.entity.VO.NwHarmonicSpectrumVO;

import com.cet.pq.pqservice.mapper.NwHarmonicDataMapper;

import com.google.common.collect.Lists;

import lombok.extern.slf4j.Slf4j;

import org.apache.commons.lang3.StringUtils;

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

import org.springframework.stereotype.Service;

import com.cet.pq.harmonicAnlysis.entity.NwHarmonicData;

import com.cet.pq.harmonicAnlysis.service.NwHarmonicDataService;

import org.springframework.transaction.annotation.Transactional;

import java.math.BigDecimal;

import java.util.ArrayList;

import java.util.List;

import java.util.Map;

import java.util.concurrent.CompletableFuture;

import java.util.concurrent.Future;

import java.util.stream.Collectors;

/\*\*

\* @Author cao lei

\* @Date 2023/1/9 18:38

\* @Version 1.0

\*/

@Service

@Slf4j

public class NwHarmonicDataServiceImpl implements NwHarmonicDataService{

@Autowired

private NwHarmonicDataMapper nwHarmonicDataMapper;

@Override

public int deleteByPrimaryKey(String id) {

return nwHarmonicDataMapper.deleteByPrimaryKey(id);

}

@Override

public int insert(NwHarmonicData record) {

return nwHarmonicDataMapper.insert(record);

}

@Override

public int insertSelective(NwHarmonicData record) {

return nwHarmonicDataMapper.insertSelective(record);

}

@Override

public NwHarmonicData selectByPrimaryKey(String id) {

return nwHarmonicDataMapper.selectByPrimaryKey(id);

}

@Override

public int updateByPrimaryKeySelective(NwHarmonicData record) {

return nwHarmonicDataMapper.updateByPrimaryKeySelective(record);

}

@Override

public int updateByPrimaryKey(NwHarmonicData record) {

return nwHarmonicDataMapper.updateByPrimaryKey(record);

}

@Override

public List<NwHarmonicDataVO> getEventList(NwHarmonicDataDTO harmonicDataDTO) {

List<NwHarmonicData> dataList = nwHarmonicDataMapper.getEventList(harmonicDataDTO);

//获取返回数据

List<NwHarmonicDataVO> list = getByType(dataList,harmonicDataDTO.getResultType());

return list;

}

@Override

public List<LineInfo> selectLineList() {

return nwHarmonicDataMapper.selectLineList();

}

@Override

@Transactional(rollbackFor = Exception.class,transactionManager = "matterhornTransactionManager")

public void saveList(List<NwHarmonicData> harmonicDataList) {

//新增

for (NwHarmonicData nwHarmonicData:harmonicDataList) {

//删除

nwHarmonicDataMapper.deleteByInfo(nwHarmonicData);

//日数据

nwHarmonicDataMapper.insertSelective(nwHarmonicData);

//月数据

String monthQuery = nwHarmonicData.getDate().substring(0,7);

List<NwHarmonicData> monthList = nwHarmonicDataMapper.selectForMonth(nwHarmonicData,monthQuery+"%");

NwHarmonicData nwHarmonicData1 = setNewData(monthList, monthQuery, 2, nwHarmonicData);

//更新

nwHarmonicDataMapper.deleteByInfo(nwHarmonicData1);

nwHarmonicDataMapper.insertSelective(nwHarmonicData1);

//年数据

String yearQuery = nwHarmonicData.getDate().substring(0,4);

List<NwHarmonicData> yearList = nwHarmonicDataMapper.selectForYear(nwHarmonicData,yearQuery+"%");

NwHarmonicData nwHarmonicData2 = setNewData(yearList, yearQuery, 3, nwHarmonicData);

//更新

nwHarmonicDataMapper.deleteByInfo(nwHarmonicData2);

nwHarmonicDataMapper.insertSelective(nwHarmonicData2);

}

}

private NwHarmonicData setNewData(List<NwHarmonicData> monthList, String date, Integer dateType,NwHarmonicData nwHarmonicData){

NwHarmonicData nwHarmonicData0 = new NwHarmonicData();

nwHarmonicData0.setId(IdUtils.fastSimpleUUID());

nwHarmonicData0.setLineId(nwHarmonicData.getLineId());

nwHarmonicData0.setLineName(nwHarmonicData.getLineName());

nwHarmonicData0.setSubstationId(nwHarmonicData.getSubstationId());

nwHarmonicData0.setSubstationName(nwHarmonicData.getSubstationName());

nwHarmonicData0.setSubstationType(nwHarmonicData.getSubstationType());

nwHarmonicData0.setType(nwHarmonicData.getType());

nwHarmonicData0.setNwm(nwHarmonicData.getNwm());

nwHarmonicData0.setDate(date);

nwHarmonicData0.setDateType(dateType);

//最值计算

//平均值

List<NwHarmonicData> collect = monthList.stream().filter(x -> x.getAAverage() != null).collect(Collectors.toList());

if (collect.size()>0){

BigDecimal divide = collect.stream().map(NwHarmonicData::getAAverage).reduce(BigDecimal.ZERO, BigDecimal::add).divide(BigDecimal.valueOf(collect.size()), 2, BigDecimal.ROUND\_HALF\_UP);

nwHarmonicData0.setAAverage(divide);

}

List<NwHarmonicData> collect0 = monthList.stream().filter(x -> x.getBAverage() != null).collect(Collectors.toList());

if (collect0.size()>0){

BigDecimal divide0 = collect0.stream().map(NwHarmonicData::getBAverage).reduce(BigDecimal.ZERO, BigDecimal::add).divide(BigDecimal.valueOf(collect0.size()), 2, BigDecimal.ROUND\_HALF\_UP);

nwHarmonicData0.setBAverage(divide0);

}

List<NwHarmonicData> collect1 = monthList.stream().filter(x -> x.getCAverage() != null).collect(Collectors.toList());

if (collect1.size()>0){

BigDecimal divide1 = collect1.stream().map(NwHarmonicData::getCAverage).reduce(BigDecimal.ZERO, BigDecimal::add).divide(BigDecimal.valueOf(collect1.size()), 2, BigDecimal.ROUND\_HALF\_UP);

nwHarmonicData0.setCAverage(divide1);

}

//最大值

List<NwHarmonicData> aMax = monthList.stream().filter(x -> x.getAMax() != null).collect(Collectors.toList());

if (aMax.size()>0){

BigDecimal max = aMax.stream().map(NwHarmonicData::getAMax).max((x1, x2) -> x1.compareTo(x2)).get();

nwHarmonicData0.setAMax(max);

}

List<NwHarmonicData> bMax = monthList.stream().filter(x -> x.getBMax() != null).collect(Collectors.toList());

if (bMax.size()>0){

BigDecimal max1 = bMax.stream().map(NwHarmonicData::getBMax).max((x1, x2) -> x1.compareTo(x2)).get();

nwHarmonicData0.setBMax(max1);

}

List<NwHarmonicData> cMax = monthList.stream().filter(x -> x.getCMax() != null).collect(Collectors.toList());

if (cMax.size()>0){

BigDecimal max2 = cMax.stream().map(NwHarmonicData::getCMax).max((x1, x2) -> x1.compareTo(x2)).get();

nwHarmonicData0.setCMax(max2);

}

//最小值

List<NwHarmonicData> aMin = monthList.stream().filter(x -> x.getAMin() != null).collect(Collectors.toList());

if (aMin.size()>0){

BigDecimal min = aMin.stream().map(NwHarmonicData::getAMin).min((x1, x2) -> x1.compareTo(x2)).get();

nwHarmonicData0.setAMin(min);

}

List<NwHarmonicData> bMin = monthList.stream().filter(x -> x.getBMin() != null).collect(Collectors.toList());

if (bMin.size()>0){

BigDecimal min1 = bMin.stream().map(NwHarmonicData::getBMin).min((x1, x2) -> x1.compareTo(x2)).get();

nwHarmonicData0.setBMin(min1);

}

List<NwHarmonicData> cMin = monthList.stream().filter(x -> x.getCMin() != null).collect(Collectors.toList());

if (cMin.size()>0){

BigDecimal min2 = cMin.stream().map(NwHarmonicData::getCMin).min((x1, x2) -> x1.compareTo(x2)).get();

nwHarmonicData0.setCMin(min2);

}

//cp95值

List<NwHarmonicData> cp = monthList.stream().filter(x -> x.getACp95()!=null).collect(Collectors.toList());

if(cp.size()>0){

BigDecimal cp95 = cp.stream().map(NwHarmonicData::getACp95).reduce(BigDecimal.ZERO, BigDecimal::add).divide(new BigDecimal(cp.size()), 2, BigDecimal.ROUND\_HALF\_UP);

nwHarmonicData0.setACp95(cp95);

}

List<NwHarmonicData> cp1 = monthList.stream().filter(x -> x.getBCp95()!=null).collect(Collectors.toList());

if (cp1.size()>0){

BigDecimal cp951 = cp1.stream().map(NwHarmonicData::getBCp95).reduce(BigDecimal.ZERO, BigDecimal::add).divide(new BigDecimal(cp1.size()), 2, BigDecimal.ROUND\_HALF\_UP);

nwHarmonicData0.setBCp95(cp951);

}

List<NwHarmonicData> cp2 = monthList.stream().filter(x -> x.getCCp95()!=null).collect(Collectors.toList());

if (cp2.size()>0){

BigDecimal cp952 = cp2.stream().map(NwHarmonicData::getCCp95).reduce(BigDecimal.ZERO, BigDecimal::add).divide(new BigDecimal(cp2.size()), 2, BigDecimal.ROUND\_HALF\_UP);

nwHarmonicData0.setCCp95(cp952);

}

return nwHarmonicData0;

}

@Override

public List<NwHarmonicSpectrumVO> getHarmonicSpectrumList(NwHarmonicDataDTO harmonicDataDTO) {

List<NwHarmonicSpectrumVO> list = new ArrayList<>();

harmonicDataDTO.setAnalyseType(1);

List<NwHarmonicDataVO> voltageList = getEventList(harmonicDataDTO);

harmonicDataDTO.setAnalyseType(2);

List<NwHarmonicDataVO> currentList = getEventList(harmonicDataDTO);

for (int i = 2; i <=25 ; i++) {

NwHarmonicSpectrumVO harmonicSpectrumVO = new NwHarmonicSpectrumVO();

harmonicSpectrumVO.setAnalyseCount(i);

//电压数据

for (NwHarmonicDataVO nw:voltageList) {

if (i==nw.getAnalyseCount().intValue()){

harmonicSpectrumVO.setVoltageA(nw.getHarmonicA());

harmonicSpectrumVO.setVoltageB(nw.getHarmonicB());

harmonicSpectrumVO.setVoltageC(nw.getHarmonicC());

//删除

voltageList.remove(nw);

break;

}

}

//电流数据

for (NwHarmonicDataVO nw:currentList) {

if (i==nw.getAnalyseCount().intValue()){

harmonicSpectrumVO.setCurrentA(nw.getHarmonicA());

harmonicSpectrumVO.setCurrentB(nw.getHarmonicB());

harmonicSpectrumVO.setCurrentC(nw.getHarmonicC());

//删除

currentList.remove(nw);

break;

}

}

list.add(harmonicSpectrumVO);

}

return list;

}

////根据所需数据类型获取数据

public List<NwHarmonicDataVO> getByType(List<NwHarmonicData> dataList,Integer type){

List<NwHarmonicDataVO> list = new ArrayList<>();

NwHarmonicDataVO harmonicDataVO = new NwHarmonicDataVO();

if (type==1){//平均值

for (NwHarmonicData nw:dataList) {

harmonicDataVO.setAnalyseCount(nw.getNwm());

harmonicDataVO.setHarmonicA(nw.getAAverage());

harmonicDataVO.setHarmonicB(nw.getBAverage());

harmonicDataVO.setHarmonicC(nw.getCAverage());

list.add(harmonicDataVO);

harmonicDataVO = new NwHarmonicDataVO();

}

}else if (type==2){//最大值

for (NwHarmonicData nw:dataList) {

harmonicDataVO.setAnalyseCount(nw.getNwm());

harmonicDataVO.setHarmonicA(nw.getAMax());

harmonicDataVO.setHarmonicB(nw.getBMax());

harmonicDataVO.setHarmonicC(nw.getCMax());

list.add(harmonicDataVO);

harmonicDataVO = new NwHarmonicDataVO();

}

}else if (type==3){//最小值

for (NwHarmonicData nw:dataList) {

harmonicDataVO.setAnalyseCount(nw.getNwm());

harmonicDataVO.setHarmonicA(nw.getAMin());

harmonicDataVO.setHarmonicB(nw.getBMin());

harmonicDataVO.setHarmonicC(nw.getCMin());

list.add(harmonicDataVO);

harmonicDataVO = new NwHarmonicDataVO();

}

}else if (type==4){//cp95值

for (NwHarmonicData nw:dataList) {

harmonicDataVO.setAnalyseCount(nw.getNwm());

harmonicDataVO.setHarmonicA(nw.getACp95());

harmonicDataVO.setHarmonicB(nw.getBCp95());

harmonicDataVO.setHarmonicC(nw.getCCp95());

list.add(harmonicDataVO);

harmonicDataVO = new NwHarmonicDataVO();

}

}

return list;

}

/\*\*

\* @param harmonicDataDTO

\* @return

\*/

@Override

public List<LineInfo> getLineList(NwHarmonicDataDTO harmonicDataDTO) {

List<LineInfo> list = Lists.newArrayList();

List<Long> cityIds = nwHarmonicDataMapper.selectCitys(harmonicDataDTO);

if (!(harmonicDataDTO.getProvinceCode()!=null&&cityIds.isEmpty())){

//市

harmonicDataDTO.setCityIds(cityIds);

List<LineInfo> lineInfos = nwHarmonicDataMapper.selectForCity(harmonicDataDTO);

list.addAll(lineInfos);

//区

List<Long> districts = nwHarmonicDataMapper.selectForDistricts(harmonicDataDTO);

harmonicDataDTO.setDistrictIds(districts);

List<LineInfo> districtLine = nwHarmonicDataMapper.selectForDistrict(harmonicDataDTO);

list.addAll(districtLine);

}

return list;

}

/\*\*

\* @param

\* @return

\*/

@Override

public List<NwHarmonicProvince> getCount() {

List<NwHarmonicProvince> provinceList = nwHarmonicDataMapper.getAllProvince();

if (!provinceList.isEmpty()){

for (NwHarmonicProvince province:provinceList) {

List<LineInfo> list = Lists.newArrayList();

//分组统计

NwHarmonicDataDTO nwHarmonicDataDTO = new NwHarmonicDataDTO();

nwHarmonicDataDTO.setProvinceCode(province.getCode());

List<LineInfo> lineList = getLineList(nwHarmonicDataDTO);

Map<Integer, Long> map = lineList.stream()

.collect(Collectors.groupingBy(LineInfo::getSubstationType, Collectors.counting()));

//谐波

LineInfo lineInfo = new LineInfo();

lineInfo.setSubstationType(SubstationTypeEnum.WAVE.getValues());

lineInfo.setSubstationTypeLabel(SubstationTypeEnum.WAVE.getName());

if (!map.isEmpty()&&map.containsKey(SubstationTypeEnum.WAVE.getValues())){

lineInfo.setCountNum(map.get(SubstationTypeEnum.WAVE.getValues()));

}else {

lineInfo.setCountNum(0L);

}

//储能

LineInfo lineInfo1 = new LineInfo();

lineInfo1.setSubstationType(SubstationTypeEnum.STORAGE.getValues());

lineInfo1.setSubstationTypeLabel(SubstationTypeEnum.STORAGE.getName());

if (!map.isEmpty()&&map.containsKey(SubstationTypeEnum.STORAGE.getValues())){ lineInfo1.setCountNum(map.get(SubstationTypeEnum.STORAGE.getValues()));

}else {

lineInfo1.setCountNum(0L);

}

LineInfo lineInfo2 = new LineInfo();

lineInfo2.setSubstationType(SubstationTypeEnum.ENERGY.getValues());

lineInfo2.setSubstationTypeLabel(SubstationTypeEnum.ENERGY.getName());

if (!map.isEmpty()&&map.containsKey(SubstationTypeEnum.ENERGY.getValues())){

lineInfo2.setCountNum(map.get(SubstationTypeEnum.ENERGY.getValues()));

}else {

lineInfo2.setCountNum(0L);

}

list.add(lineInfo);

list.add(lineInfo1);

list.add(lineInfo2);

province.setLineInfos(list);

}

}

return provinceList;

}

@Override

public NwHarmonicSpectrumVO selectDetailData(NwHarmonicDataDTO harmonicDataDTO) {

NwHarmonicSpectrumVO result = new NwHarmonicSpectrumVO();

harmonicDataDTO.setAnalyseType(3);

NwHarmonicData vo = nwHarmonicDataMapper.selectVoltage(harmonicDataDTO);

harmonicDataDTO.setAnalyseType(4);

NwHarmonicData vo2 = nwHarmonicDataMapper.selectVoltage(harmonicDataDTO);

Integer resultType = harmonicDataDTO.getResultType();

if (1==resultType.intValue()){

result.setVoltageA(vo!=null?vo.getAAverage():null);

result.setVoltageB(vo!=null?vo.getBAverage():null);

result.setVoltageC(vo!=null?vo.getCAverage():null);

result.setCurrentA(vo2!=null?vo2.getAAverage():null);

result.setCurrentB(vo2!=null?vo2.getBAverage():null);

result.setCurrentC(vo2!=null?vo2.getCAverage():null);

}else if(2==resultType.intValue()){

result.setVoltageA(vo!=null?vo.getAMax():null);

result.setVoltageB(vo!=null?vo.getBMax():null);

result.setVoltageC(vo!=null?vo.getCMax():null);

result.setCurrentA(vo2!=null?vo2.getAMax():null);

result.setCurrentB(vo2!=null?vo2.getBMax():null);

result.setCurrentC(vo2!=null?vo2.getCMax():null);

}else if(3==resultType.intValue()){

result.setVoltageA(vo!=null?vo.getAMin():null);

result.setVoltageB(vo!=null?vo.getBMin():null);

result.setVoltageC(vo!=null?vo.getCMin():null);

result.setCurrentA(vo2!=null?vo2.getAMin():null);

result.setCurrentB(vo2!=null?vo2.getBMin():null);

result.setCurrentC(vo2!=null?vo2.getCMin():null);

}else if (4==resultType.intValue()){

result.setVoltageA(vo!=null?vo.getACp95():null);

result.setVoltageB(vo!=null?vo.getBCp95():null);

result.setVoltageC(vo!=null?vo.getCCp95():null);

result.setCurrentA(vo2!=null?vo2.getCAverage():null);

result.setCurrentB(vo2!=null?vo2.getBCp95():null);

result.setCurrentC(vo2!=null?vo2.getCCp95():null);

}

return result;

}

}

import com.cet.pq.common.model.Result;

import com.cet.pq.common.model.datalog.DataLogData;

import com.cet.pq.common.model.datalog.TrendDataVo;

import com.cet.pq.common.model.datalog.TrendSearchListVo;

import com.cet.pq.common.model.datalog.TrendSearchVo;

import com.cet.pq.common.time.DateUtils;

import com.cet.pq.common.utils.HttpUtils;

import com.cet.pq.common.utils.JsonTransferUtils;

import com.cet.pq.common.utils.ParseDataUtil;

import com.cet.pq.pqservice.model.victoriametrics.MetricResult;

import com.cet.pq.pqservice.model.victoriametrics.VMDataResult;

import com.cet.pq.pqservice.model.victoriametrics.VictoriaMetricsParam;

import com.cet.pq.pqservice.model.victoriametrics.VictoriaMetricsResult;

import com.cet.pq.pqservice.service.VictoriaMetricsService;

import org.apache.commons.collections4.CollectionUtils;

import org.apache.commons.lang3.StringUtils;

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

import org.springframework.stereotype.Service;

import java.text.ParseException;

import java.util.\*;

import java.util.stream.Collectors;

/\*\*

\* @author gongtong

\* @date 2021/7/29 9:24

\* @description

\*/

@Service

public class VictoriaMetricsServiceImpl implements VictoriaMetricsService {

@Value("${victoriametrics.url}")

public String victoriaMetricsUrl;

@Override

public Result<List<TrendDataVo>> queryTrendCurveData(TrendSearchListVo searchVo, Boolean fill, Integer dataLogInterval) {

VictoriaMetricsParam victoriaMetricsParam = new VictoriaMetricsParam();

//组装vm查询参数

victoriaMetricsParam.setEnd(DateUtils.stringToDate(searchVo.getEndTime(), DateUtils.DATE\_TO\_STRING\_DETAIAL\_PATTERN).getTime() / 1000);

victoriaMetricsParam.setStart(DateUtils.stringToDate(searchVo.getStartTime(), DateUtils.DATE\_TO\_STRING\_DETAIAL\_PATTERN).getTime() / 1000);

victoriaMetricsParam.setStep(searchVo.getInterval() \* 60);

List<TrendSearchVo> meterConfigs = searchVo.getMeterConfigs();

//组装cet\_value参数

VictoriaMetricsParam.CetValue query = new VictoriaMetricsParam.CetValue();

Set<String> dataIds = meterConfigs.stream().map(t -> ParseDataUtil.parseString(t.getDataId())).collect(Collectors.toSet());

Set<String> dataTypeIds = meterConfigs.stream().map(t -> ParseDataUtil.parseString(t.getDataTypeId())).collect(Collectors.toSet());

Set<String> deviceIds = meterConfigs.stream().map(t -> ParseDataUtil.parseString(t.getDeviceId())).collect(Collectors.toSet());

Set<String> logicalIds = meterConfigs.stream().map(t -> ParseDataUtil.parseString(t.getLogicalId())).collect(Collectors.toSet());

query.setDataId("~'" + String.join("|", dataIds) + "'");

query.setDeviceId("~'" + String.join("|", deviceIds) + "'");

query.setDataTypeId("~'" + String.join("|", dataTypeIds) + "'");

query.setLogicalId("~'" + String.join("|", logicalIds) + "'");

//组装查询参数

Map<String, String> params = new HashMap<>(16);

Map<String, String> headers = new HashMap<>(16);

String queryVmUrl = victoriaMetricsUrl + "/api/v1/export";

params.put("match", query.toString());

params.put("start", ParseDataUtil.parseString(victoriaMetricsParam.getStart()));

params.put("end", ParseDataUtil.parseString(victoriaMetricsParam.getEnd()));

String result = HttpUtils.get(queryVmUrl, "UTF-8", params, headers);

if (StringUtils.isBlank(result)) {

return new Result<>();

}

List<TrendDataVo> trendDataVos = new ArrayList<>();

//转换json标准格式

result = result.replace("\n", ",");

result = "[" + result.substring(0, result.length() - 1) + "]";

List<MetricResult> metricResults = JsonTransferUtils.transferJsonString(result, MetricResult.class);

//拼接结果值

metricResults.forEach(metricResult -> {

MetricResult.Metric metric = metricResult.getMetric();

TrendDataVo trendDataVo = new TrendDataVo();

trendDataVo.setDeviceId(ParseDataUtil.parseLong(metric.getDevId()));

trendDataVo.setDataId(ParseDataUtil.parseLong(metric.getDataId()));

trendDataVo.setDataTypeId(ParseDataUtil.parseInteger(metric.getDataTypeId()));

trendDataVo.setLogicalId(ParseDataUtil.parseInteger(metric.getLogicalId()));

List<DataLogData> dataLogDataList = new ArrayList<>();

Double[] values = metricResult.getValues();

Long[] timestamps = metricResult.getTimestamps();

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

DataLogData dataLogData = new DataLogData();

dataLogData.setTime(ParseDataUtil.parseLong(timestamps[i]));

dataLogData.setValue(ParseDataUtil.parseDouble(values[i]));

dataLogData.setStatus(0);

dataLogDataList.add(dataLogData);

}

// 填充数据

List<DataLogData> dataLogDataList1 = dataLogDataList;

if (fill) {

Integer interval = getInterval(dataLogDataList, searchVo.getInterval());

if (interval != null) {

dataLogDataList1 = fillData(DateUtils.stringToDate(searchVo.getStartTime(), DateUtils.DATE\_TO\_STRING\_DETAIAL\_PATTERN).getTime(),

DateUtils.stringToDate(searchVo.getEndTime(), DateUtils.DATE\_TO\_STRING\_DETAIAL\_PATTERN).getTime(),

dataLogDataList, interval);

}

}

trendDataVo.setDataList(dataLogDataList1);

trendDataVos.add(trendDataVo);

});

return Result.success(trendDataVos);

}

/\*\*

\* @Description: 填充数据

\* @return: java.util.List<com.cet.pq.common.model.datalog.DataLogData>

\*\*/

private List<DataLogData> fillData(Long starTime, Long endTime, List<DataLogData> dataLogValueList, Integer tempInterval) {

Map<Long, DataLogData> timeValueMap = dataLogValueList.stream().collect(Collectors.toMap(DataLogData::getTime, x -> x, (v1, v2) -> v1));

while (starTime < endTime) {

if (!timeValueMap.containsKey(starTime)) {

DataLogData dataLogData = new DataLogData();

dataLogData.setStatus(0);

dataLogData.setTime(starTime);

timeValueMap.put(starTime, dataLogData);

}

starTime += tempInterval;

}

return timeValueMap.values().stream().sorted(Comparator.comparing(DataLogData::getTime)).collect(Collectors.toList());

}

/\*\*

\* 获取数据间隔，单位毫秒

\* @return java.lang.Integer

\* @date 2020/8/26 11:00

\*/

private Integer getInterval(List<DataLogData> dataLogValueList, Integer interval) {

if (interval != null && interval > 0) {

return interval \* 60 \* 1000;

}

// 自动补点，数据条数需要大于2

int autoFillMinValueCount = 3;

if (dataLogValueList == null || dataLogValueList.size() < autoFillMinValueCount) {

return null;

}

Map<Integer, Integer> intervalCountMap = new HashMap<>(10);

for (int i = 1; i < dataLogValueList.size(); i++) {

int tempInterval = (int) (dataLogValueList.get(i).getTime() - dataLogValueList.get(i - 1).getTime());

if (!intervalCountMap.containsKey(tempInterval)) {

intervalCountMap.put(tempInterval, 0);

}

intervalCountMap.put(tempInterval, intervalCountMap.get(tempInterval) + 1);

}

int dataInterval = 0, intervalCount = 0;

for (Map.Entry<Integer, Integer> item : intervalCountMap.entrySet()) {

if (item.getValue() > intervalCount) {

intervalCount = item.getValue();

dataInterval = item.getKey();

} else if (item.getValue().equals(intervalCount)) {

if (item.getKey() < dataInterval) {

dataInterval = item.getKey();

}

}

}

return dataInterval;

}

}

package com.ixunqi.dev.service.impl;

import com.ixunqi.dev.mapper.HarmonicParamMapper;

import com.ixunqi.dev.model.HarmonicParam;

import com.ixunqi.dev.model.vo.Result;

import com.ixunqi.dev.service.HarmonicParamService;

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

import org.springframework.stereotype.Service;

import java.util.List;

/\*\*

\* @author chenheng

\* @date 2023/8/15 15:47

\* @param

\*/

@Service

public class HarmonicParamServiceImpl implements HarmonicParamService {

@Autowired

private HarmonicParamMapper harmonicParamMapper;

@Override

public HarmonicParam getHarmonicParam() {

List<HarmonicParam> harmonicParamList= harmonicParamMapper.getHarmonicParam(0);

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

return harmonicParamList.get(0);

}

return null;

}

@Override

public HarmonicParam getLockHarmonicParam() {

List<HarmonicParam> harmonicParamList= harmonicParamMapper.getHarmonicParam(1);

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

return harmonicParamList.get(0);

}

return null;

}

/\*\*

\* @author chenheng

\* 保存数据并调用算法生成结果

\* @date 2023/8/22 17:52

\* @param

\*/

@Override

public Result saveHarmonicParam(HarmonicParam powerGridParam) {

//保存数据

harmonicParamMapper.updateByIslock(powerGridParam);

//调用算法生成结果

return Result.success("1");

}

}

package com.ixunqi.dev.service.impl;  
import com.ixunqi.dev.mapper.ConverterStationParamMapper;  
import com.ixunqi.dev.model.ConverterStationParam;  
import com.ixunqi.dev.model.vo.Result;  
import com.ixunqi.dev.service.ConverterStationParamService;  
import org.springframework.beans.factory.annotation.Autowired;  
import org.springframework.stereotype.Service;  
import java.util.List;  
/\*\*  
 \* @author chenheng  
 \* @date 2023/8/15 15:47  
 \* @param  
 \*/  
@Service  
public class ConverterStationParamServiceImpl implements ConverterStationParamService {  
 @Autowired  
 private ConverterStationParamMapper converterStationParamMapper;  
 @Override  
 public ConverterStationParam getConverterStationParam() {  
 List<ConverterStationParam> converterStationParamList= converterStationParamMapper.getConverterStationParam(0);  
 if (null!=converterStationParamList&&converterStationParamList.size()>0){  
 return converterStationParamList.get(0);  
 }  
 return null;  
 }  
 @Override  
 public ConverterStationParam getLockConverterStationParam() {  
 List<ConverterStationParam> converterStationParamList= converterStationParamMapper.getConverterStationParam(1);  
 if (null!=converterStationParamList&&converterStationParamList.size()>0){  
 return converterStationParamList.get(0);  
 }  
 return null;  
 }  
 /\*\*  
 \* @author chenheng  
 \* 保存数据并调用算法生成结果  
 \* @date 2023/8/22 17:52  
 \* @param  
 \*/  
 @Override  
 public Result saveConverterStationParam(ConverterStationParam converterStationParam) {  
 //保存数据  
converterStationParamMapper.updateByIslock(converterStationParam);  
 //调用算法生成结果  
 return Result.success("1");  
 }  
}

package com.hungraim.wx.httpinvoker.impl;  
import java.lang.reflect.InvocationTargetException;  
import java.util.ArrayList;  
import java.util.Date;  
import java.util.List;  
import java.util.regex.Pattern;  
import javax.annotation.Resource;  
impot org.apache.commons.beanutils.BeanUtils;  
import org.apache.commons.beanutils.PropertyUtils;  
import org.apache.commons.lang3.StringUtils;  
import org.slf4j.Logger;  
import org.springframework.stereotype.Component;  
import org.springframework.web.util.HtmlUtils;  
import com.hungraim.wx.base.page.PageParam;  
import com.hungraim.wx.base.service.CommonService;  
import com.hungraim.wx.base.util.LoggerUtil;  
import com.hungraim.wx.common.constant.SysConstant;  
import com.hungraim.wx.common.vo.StatefulRtnBase;  
import com.hungraim.wx.component.services.model.BusinessActivities;  
import com.hungraim.wx.component.services.model.ItspActivitiesTask;  
import com.hungraim.wx.component.services.service.ActivitiesTaskService;  
import com.hungraim.wx.component.services.service.BusinessActivitiesService;  
import com.hungraim.wx.httpinvoker.inf.IActivitiesTaskService;  
import com.hungraim.wx.httpinvoker.vo.BusinessActivitiesVo;  
import com.hungraim.wx.httpinvoker.vo.IUserInfoVo;  
import com.hungraim.wx.httpinvoker.vo.ItspActivitiesTaskVo;  
import com.hungraim.wx.httpinvoker.vo.SysFileVo;  
import com.hungraim.wx.system.model.SysFile;  
import com.hungraim.wx.system.service.SysFileService;  
  
@Component  
public class ActivitiesTaskServiceImpl implements IActivitiesTaskService {  
 Logger logger = LoggerUtil.getLogger();  
 @Resource  
 private ActivitiesTaskService activitiesTaskService;  
 @Resource  
 private BusinessActivitiesService businessActivitiesService;  
 @Resource  
 private SysFileService sysFileService;  
 @Override  
 public StatefulRtnBase deleteActivitiesTaskServiceById(String id) {  
 StatefulRtnBase statefulRtnBase = new StatefulRtnBase();  
 boolean b = activitiesTaskService.deleteActTaskById(id);  
 if (!b) {  
 logger.warn("业务活动准备任务删除失败");  
 setStatefulRtnBase(statefulRtnBase, "业务活动准备任务删除失败");  
 }  
 return statefulRtnBase;  
 }  
 @Override  
 public StatefulRtnBase saveOrUpdate(ItspActivitiesTaskVo itspActivitiesTaskVo, IUserInfoVo currentUser,  
 String[] files) {  
 StatefulRtnBase statefulRtnBase = new StatefulRtnBase();  
 //判断当前操作人是否为空  
 if (null != currentUser) {  
 String curUserName = currentUser.getUserName();  
 //判断传入的vo是否为空  
 if (null != itspActivitiesTaskVo) {  
 ItspActivitiesTask itspActivitiesTask = new ItspActivitiesTask();  
 if (StringUtils.isNotBlank(itspActivitiesTaskVo.getId())) {  
 //根据id获取活动  
 itspActivitiesTask = activitiesTaskService.getActivitiesTaskById(itspActivitiesTaskVo.getId());  
 } else {  
 itspActivitiesTask = new ItspActivitiesTask();  
 itspActivitiesTask.setCrtDttm(new Date());  
 itspActivitiesTask.setTaskBusNo(activitiesTaskService.getTaskBusNo("RW", CommonService.SEQ\_ACTIVITIESTASK\_BUSNO,itspActivitiesTaskVo.getBusinessActivitiesVo().getBusNo()));  
 }  
// itspActivitiesTask.setTaskName(itspActivitiesTaskVo.getTaskName());//任务名称  
 if(StringUtils.isNotBlank(itspActivitiesTaskVo.getResponsibleDept())){  
 itspActivitiesTask.setResponsibleDept(itspActivitiesTaskVo.getResponsibleDept());//负责部门  
 }  
 if(StringUtils.isNotBlank(itspActivitiesTaskVo.getResponsibleUser())){  
 itspActivitiesTask.setResponsibleUser(itspActivitiesTaskVo.getResponsibleUser());//负责人  
 }  
 if(null != itspActivitiesTaskVo.getPlanTime()){  
 itspActivitiesTask.setPlanTime(itspActivitiesTaskVo.getPlanTime());//计划完成时间  
 }  
 if(null != itspActivitiesTaskVo.getActualTime()){  
 itspActivitiesTask.setActualTime(itspActivitiesTaskVo.getActualTime());//实际完成时间  
 }  
 if(null != itspActivitiesTaskVo.getTaskStatus()){  
 itspActivitiesTask.setTaskStatus(itspActivitiesTaskVo.getTaskStatus());  
 }  
 if(StringUtils.isNotBlank(itspActivitiesTaskVo.getNote())){  
 itspActivitiesTask.setNote(itspActivitiesTaskVo.getNote());  
 }  
 itspActivitiesTask.setLastUpdateUserName(curUserName);  
 itspActivitiesTask.setLastUpdateDttm(new Date());//最后更新时间   
 itspActivitiesTask.getFiles().clear();  
 activitiesTaskService.saveOrUpdate(itspActivitiesTask);  
 //关联附件  
 linkActFiles(itspActivitiesTask,files);  
 } else {  
 logger.warn("传入的vo为空");  
 setStatefulRtnBase(statefulRtnBase, "传入的vo为空");  
 }  
 } else {   
 logger.warn("操作人为空确认失败");  
 setStatefulRtnBase(statefulRtnBase, "操作人为空确认失败");  
 }  
 return statefulRtnBase;  
 }  
 /\*\*  
 \* @param statefulRtnBase  
 \* @param errorMsg  
 \*/  
 private void setStatefulRtnBase(StatefulRtnBase statefulRtnBase, String errorMsg) {  
 statefulRtnBase.setReqRtnStatus(false);  
 statefulRtnBase.setErrorCode(statefulRtnBase.ERRCODE\_FAIL);  
 statefulRtnBase.setErrorMsg(errorMsg);  
 }  
 @Override  
 public ItspActivitiesTaskVo getActivitiesTaskById(String id) {  
 ItspActivitiesTaskVo itspActivitiesTaskVo = new ItspActivitiesTaskVo();  
 ItspActivitiesTask itspActivitiesTask = activitiesTaskService.getActivitiesTaskById(id);  
 try {  
 PropertyUtils.copyProperties(itspActivitiesTaskVo, itspActivitiesTask);  
 BusinessActivitiesVo businessActivitiesVo = new BusinessActivitiesVo();  
 BeanUtils.copyProperties(businessActivitiesVo, itspActivitiesTask.getBusinessActivities());  
 List<SysFileVo> sysFileVoList = new ArrayList<SysFileVo>();  
 for (SysFile sysFile : itspActivitiesTask.getBusinessActivities().getFiles()) {  
 SysFileVo sysFileVo = new SysFileVo();  
 PropertyUtils.copyProperties(sysFileVo, sysFile);  
 sysFileVoList.add(sysFileVo);  
 }  
 businessActivitiesVo.setFiles(sysFileVoList);  
 List<SysFileVo> sysFileVoList2 = new ArrayList<SysFileVo>();  
 for (SysFile sysFile : itspActivitiesTask.getFiles()) {  
 SysFileVo sysFileVo = new SysFileVo();  
 PropertyUtils.copyProperties(sysFileVo, sysFile);  
 sysFileVoList2.add(sysFileVo);  
 }  
 itspActivitiesTaskVo.setFiles(sysFileVoList2);  
 itspActivitiesTaskVo.setBusinessActivitiesVo(businessActivitiesVo);  
 } catch (NoSuchMethodException e) {  
 logger.error("属性的IllegalAccessException问题", e);  
 } catch (IllegalAccessException e) {  
 logger.error("属性的IllegalAccessException问题", e);  
 } catch (InvocationTargetException e) {  
 logger.error("属性的IllegalAccessException问题", e);  
 }  
  
 return itspActivitiesTaskVo;  
 }  
 @Override  
 public List<ItspActivitiesTaskVo> checkActTask(String id) {  
 List<ItspActivitiesTaskVo> listVo = new ArrayList<ItspActivitiesTaskVo>();  
 List<ItspActivitiesTask> list = activitiesTaskService.checkActTask(id);  
 if (null != list && list.size() > 0) {  
 for (ItspActivitiesTask itspActivitiesTask : list) {  
 ItspActivitiesTaskVo itspActivitiesTaskVo = new ItspActivitiesTaskVo();  
 try {  
 BeanUtils.copyProperties(itspActivitiesTaskVo, itspActivitiesTask);  
 List<SysFileVo> sysFileVoList = new ArrayList<SysFileVo>();  
 for (SysFile sysFile : itspActivitiesTask.getFiles()) {  
 SysFileVo sysFileVo = new SysFileVo();  
 PropertyUtils.copyProperties(sysFileVo, sysFile);  
 sysFileVoList.add(sysFileVo);  
 }  
 itspActivitiesTaskVo.setFiles(sysFileVoList);  
 } catch (NoSuchMethodException e) {  
 logger.error("属性的IllegalAccessException问题", e);  
 } catch (IllegalAccessException e) {  
 e.printStackTrace();  
 } catch (InvocationTargetException e) {  
 e.printStackTrace();  
 }  
 listVo.add(itspActivitiesTaskVo);  
 }  
 }  
 return listVo;  
 }  
 @Override  
 public PageParam<ItspActivitiesTaskVo> findActTaskList(PageParam<ItspActivitiesTaskVo> pageResult, String busNo,  
 String title, String taskName, String responsibleDept, String responsibleUser,String taskStatus,String taskBusNo) {  
 List<ItspActivitiesTaskVo> listVo = new ArrayList<ItspActivitiesTaskVo>();  
 PageParam<ItspActivitiesTask> pageParam = activitiesTaskService.findAllActivitiesTask(pageResult,  
 busNo, title, taskName, responsibleDept, responsibleUser,taskStatus,taskBusNo);  
 PageParam<ItspActivitiesTaskVo> pagerVo = new PageParam<ItspActivitiesTaskVo>(pageParam);  
 if (null != pageParam && null != pageParam.getPageList() && pageParam.getPageList().size() > 0) {  
 for (ItspActivitiesTask itspActivitiesTask : pageParam.getPageList()) {  
 ItspActivitiesTaskVo itspActivitiesTaskVo1 = new ItspActivitiesTaskVo();  
 try {  
 BeanUtils.copyProperties(itspActivitiesTaskVo1, itspActivitiesTask);  
 List<SysFileVo> sysFileVoList2 = new ArrayList<SysFileVo>();  
 for (SysFile sysFile : itspActivitiesTask.getFiles()) {  
 SysFileVo sysFileVo = new SysFileVo();  
 PropertyUtils.copyProperties(sysFileVo, sysFile);  
 sysFileVoList2.add(sysFileVo);  
 }  
 itspActivitiesTaskVo1.setFiles(sysFileVoList2);  
 BusinessActivitiesVo businessActivitiesVo = new BusinessActivitiesVo();  
 BeanUtils.copyProperties(businessActivitiesVo, itspActivitiesTask.getBusinessActivities());  
 List<SysFileVo> sysFileVoList = new ArrayList<SysFileVo>();  
 for (SysFile sysFile : itspActivitiesTask.getBusinessActivities().getFiles()) {  
 SysFileVo sysFileVo = new SysFileVo();  
 PropertyUtils.copyProperties(sysFileVo, sysFile);  
 sysFileVoList.add(sysFileVo);  
 }  
 businessActivitiesVo.setFiles(sysFileVoList);  
 itspActivitiesTaskVo1.setBusinessActivitiesVo(businessActivitiesVo);  
 } catch (NoSuchMethodException e) {  
 logger.error("属性的IllegalAccessException问题", e);  
 } catch (IllegalAccessException e) {  
 e.printStackTrace();  
 } catch (InvocationTargetException e) {  
 e.printStackTrace();  
 }  
 listVo.add(itspActivitiesTaskVo1);  
 }  
 pagerVo.setPageList(listVo);  
 }  
 return pagerVo;  
 }  
 @Override  
 public List<SysFileVo> saveSysFile(List<SysFileVo> sysFileVoList, IUserInfoVo userInfoVo) {  
 if (sysFileVoList.size() > 0) {  
 for (SysFileVo sysFileVo : sysFileVoList) {  
 SysFile file = new SysFile();  
 try {  
 PropertyUtils.copyProperties(file, sysFileVo);  
 file.setCrtDttm(new Date());  
 file.setLastUpdateDttm(new Date());  
 file.setEnableFlag(true);  
 file.setCrtUser(userInfoVo.getUserName());  
 file.setLastUpdateUser(userInfoVo.getUserName());  
 //解决xss漏洞  
 String regex = SysConstant.REGEX\_HTML();  
 String fileName = sysFileVo.getFileName();  
 if (StringUtils.isNotBlank(fileName) && Pattern.matches(regex, fileName)) {  
 fileName = HtmlUtils.htmlEscapeDecimal(fileName);  
 }  
 file.setFileName(fileName);  
 sysFileService.saveOrUpdateSysFile(file);  
 } catch (NoSuchMethodException e) {  
 logger.error("属性的IllegalAccessException问题", e);  
 } catch (IllegalAccessException e) {  
 logger.error("属性的IllegalAccessException问题", e);  
 } catch (InvocationTargetException e) {  
 logger.error("属性的IllegalAccessException问题", e);  
 }  
 }  
 }  
 return sysFileVoList;  
 }  
 @Override  
 public SysFileVo downLoadFile(String fileId) {  
 SysFileVo sysFileVo = new SysFileVo();  
 if (fileId != null) {  
 SysFile sysFile = sysFileService.findSysFile(fileId);  
 try {  
 BeanUtils.copyProperties(sysFileVo, sysFile);  
 } catch (IllegalAccessException e) {  
 e.printStackTrace();  
 } catch (InvocationTargetException e) {  
 e.printStackTrace();  
 }  
 }  
 return sysFileVo;  
 }  
 private void linkActFiles(ItspActivitiesTask itspActivitiesTask,String[] files) {  
 if (StringUtils.isNotBlank(itspActivitiesTask.getId())) {  
 if (null != files && files.length > 0) {  
 for (int i = 0; i < files.length; i++) {  
 sysFileService.updateActTaskSysFileSql(new Object[] { itspActivitiesTask.getId(), files[i] });  
 }  
 }  
 }  
 //commonService.flush();  
 }  
 @Override  
 public PageParam<BusinessActivitiesVo> findAllBusinessActivities(PageParam<BusinessActivitiesVo> pageResult,  
 String startTime, String endTime, String busNo, String ftPlanningDept, String status) {  
 List<BusinessActivitiesVo> businessActivitiesListVo = new ArrayList<BusinessActivitiesVo>();  
 PageParam<BusinessActivitiesVo> pagerVo = new PageParam<BusinessActivitiesVo>();  
 PageParam<BusinessActivities> pager = activitiesTaskService.findAllBusinessActivities(pageResult, startTime, endTime, busNo, ftPlanningDept, status);  
 if (null != pager && pager.getPageList().size() > 0) {  
 for (BusinessActivities businessActivities : pager.getPageList()) {  
 BusinessActivitiesVo newBusinessActivitiesVo = new BusinessActivitiesVo();  
 try {  
 BeanUtils.copyProperties(newBusinessActivitiesVo, businessActivities);  
 List<SysFileVo> sysFileVoList = new ArrayList<SysFileVo>();  
 for (SysFile sysFile : businessActivities.getFiles()) {  
 SysFileVo sysFileVo = new SysFileVo();  
 PropertyUtils.copyProperties(sysFileVo, sysFile);  
 sysFileVoList.add(sysFileVo);  
 }  
 newBusinessActivitiesVo.setFiles(sysFileVoList);  
 } catch (NoSuchMethodException e) {  
 logger.error("属性的IllegalAccessException问题", e);  
 } catch (IllegalAccessException e) {  
 logger.error("属性的IllegalAccessException问题", e);  
 } catch (InvocationTargetException e) {  
 logger.error("属性的IllegalAccessException问题", e);  
 }  
 businessActivitiesListVo.add(newBusinessActivitiesVo);  
 }  
 }  
 pagerVo.setPageList(businessActivitiesListVo);  
 pagerVo.setPageNo(pager.getPageNo());  
 pagerVo.setPageSize(pager.getPageSize());  
 pagerVo.setTotalSize(pager.getTotalSize());  
 return pagerVo;  
 }  
 @Override  
 public void saveOrUpdate(ItspActivitiesTaskVo itspActivitiesTaskVo,String businessActId) {  
 if (null != itspActivitiesTaskVo) {  
 BusinessActivities businessActivities = businessActivitiesService.findActivitieById(businessActId);  
 try {  
 ItspActivitiesTask itspActivitiesTask = new ItspActivitiesTask();  
 BeanUtils.copyProperties(itspActivitiesTask, itspActivitiesTaskVo);  
 itspActivitiesTask.setTaskBusNo(activitiesTaskService.getTaskBusNo("RW", CommonService.SEQ\_ACTIVITIESTASK\_BUSNO,businessActivities.getBusNo()));  
 if (StringUtils.isNotBlank(businessActId)) {  
 itspActivitiesTask.setBusinessActivities(businessActivities);  
 }  
 activitkService.saveOrUpdate(itspActivitiesTask);  
 } catch (IllegalAccessException e) {  
 e.printStackTrace();  
 } catch (InvocationTargetException e) {  
 e.printStackTrace();  
 }  
 }  
 }  
 @Override  
 public void deleteActTaskByBusId(String businessActId) {  
 activitiesTaskService.deleteFkidSql(new Object[] {businessActId});  
 }  
 @Override  
 public List<ItspActivitiesTaskVo> findActTaskByBusNo(String busNo) {  
 List<ItspActivitiesTask> itspActivitiesTaskList = activitiesTaskService.findActTaskByBusNo(busNo);  
 List<ItspActivitiesTaskVo> listVo = new ArrayList<ItspActivitiesTaskVo>();  
 if (null != itspActivitiesTaskList && itspActivitiesTaskList.size() > 0) {  
 for (ItspActivitiesTask itspActivitiesTask : itspActivitiesTaskList) {  
 ItspActivitiesTaskVo itspActivitiesTaskVo1 = new ItspActivitiesTaskVo();  
 try {  
 BeanUtils.copyProperties(itspActivitiesTaskVo1, itspActivitiesTask);  
 List<SysFileVo> sysFileVoList2 = new ArrayList<SysFileVo>();  
 for (SysFile sysFile : itspActivitiesTask.getFiles()) {  
 SysFileVo sysFileVo = new SysFileVo();  
 PropertyUtils.copyProperties(sysFileVo, sysFile);  
 sysFileVoList2.add(sysFileVo);  
 }  
 itspActivitiesTaskVo1.setFiles(sysFileVoList2);  
 } catch (NoSuchMethodException e) {  
 logger.error("属性的IllegalAccessException问题", e);  
 } catch (IllegalAccessException e) {  
 e.printStackTrace();  
 } catch (InvocationTargetException e) {  
 e.printStackTrace();  
 }  
 listVo.add(itspActivitiesTaskVo1);  
 }  
 }  
 return listVo;  
 }  
 @Override  
 public void delRelation(String id) {  
 if (StringUtils.isNotBlank(id)) {  
 sysFileService.updateActTaskSysFileSql(new Object[] { null, id });  
 }  
 }  
}  
package com.hungraim.wx.httpinvoker.impl;  
import java.lang.reflect.InvocationTargetException;  
import java.util.ArrayList;  
import java.util.Date;  
import java.util.HashMap;  
import java.util.List;  
import java.util.Map;  
import javax.annotation.Resource;  
import org.apache.commons.beanutils.BeanUtils;  
import org.apache.commons.lang.StringUtils;  
import org.slf4j.Logger;  
import org.springframework.stereotype.Component;  
import com.hungraim.wx.base.WechatBusinessException;  
import com.hungraim.wx.base.page.PageParam;  
import com.hungraim.wx.base.util.DateUtils;  
import com.hungraim.wx.base.util.LoggerUtil;  
import com.hungraim.wx.base.util.WechatITSPUtil;  
import com.hungraim.wx.common.constant.SysConstant;  
import com.hungraim.wx.common.enums.CheckIncType;  
import com.hungraim.wx.common.enums.DateParam;  
import com.hungraim.wx.common.enums.FaultPriorityType;  
import com.hungraim.wx.common.enums.ItspApprovalStatus;  
import com.hungraim.wx.common.enums.ItspServiceSourceSys;  
import com.hungraim.wx.common.enums.ItspServiceStatus;  
import com.hungraim.wx.common.enums.ItspServicesType;  
import com.hungraim.wx.common.enums.LoggerType;  
import com.hungraim.wx.common.enums.NeedLevel;  
import com.hungraim.wx.common.enums.PriorityType;  
import com.hungraim.wx.common.enums.RiskLevel;  
import com.hungraim.wx.common.enums.SubmitType;  
import com.hungraim.wx.common.enums.SyncStatus;  
import com.hungraim.wx.common.enums.ToDoStatus;  
import com.hungraim.wx.common.enums.ToDoTier;  
import com.hungraim.wx.common.enums.ToDoType;  
import com.hungraim.wx.common.vo.StatefulRtnBase;  
import com.hungraim.wx.component.services.model.ItspBMCJudgeCondition;  
import com.hungraim.wx.component.services.model.ItspChange;  
import com.hungraim.wx.component.services.model.ItspIncident;  
import com.hungraim.wx.component.services.model.ItspNeed;  
import com.hungraim.wx.component.services.model.ItspRelease;  
import com.hungraim.wx.component.services.model.ItspRequest;  
import com.hungraim.wx.component.services.model.ItspServices;  
import com.hungraim.wx.component.services.model.ItspServicesApproval;  
import com.hungraim.wx.component.services.model.ItspServicesApprovalConfig;  
import com.hungraim.wx.component.services.model.ItspServicesCatalogMapping;  
import com.hungraim.wx.component.services.model.ItspServicesOperAudit;  
import com.hungraim.wx.component.services.service.ApprovalConfigService;  
import com.hungraim.wx.component.services.service.IncidentService;  
import com.hungraim.wx.component.services.service.ItspBMCJudgeConditionService;  
import com.hungraim.wx.component.services.service.ItspNeedService;  
import com.hungraim.wx.component.services.service.ItspServicesApprovalService;  
import com.hungraim.wx.component.services.service.ItspServicesCatalogMappingService;  
import com.hungraim.wx.component.services.service.ItspServicesService;  
import com.hungraim.wx.component.services.service.OperAuditService;  
import com.hungraim.wx.component.todoitems.model.ToDoItems;  
import com.hungraim.wx.component.todoitems.service.ToDoItemsService;  
import com.hungraim.wx.httpinvoker.inf.IApprovalService;  
import com.hungraim.wx.httpinvoker.utils.ToDoUtils;  
import com.hungraim.wx.httpinvoker.vo.IToDoItemsVo;  
import com.hungraim.wx.httpinvoker.vo.IUserInfoVo;  
import com.hungraim.wx.httpinvoker.vo.ItspBMCJudgeConditionVo;  
import com.hungraim.wx.httpinvoker.vo.ItspIncidentVo;  
import com.hungraim.wx.httpinvoker.vo.ItspRequestWorkOrderVo;  
import com.hungraim.wx.httpinvoker.vo.ItspServicesVo;  
import com.hungraim.wx.system.model.User;  
import com.hungraim.wx.system.model.UserGroup;  
import com.hungraim.wx.system.model.UserGroupMembers;  
import com.hungraim.wx.system.service.LoggerRecordService;  
import com.hungraim.wx.system.service.OrganService;  
import com.hungraim.wx.system.service.UserService;  
import com.hungraim.wx.third.bmc.services.incident.IncidentServices;  
import com.hungraim.wx.third.bmc.services.request.RequestApprovalSync;  
import com.hungraim.wx.third.prob.services.PROBIncidentSync;  
import com.hungraim.wx.third.vp.need.CntpVpNeed;  
import com.hungraim.wx.third.yxtp.ToDoNum;  
@Component  
public class ApprovalServiceImpl implements IApprovalService {  
 Logger logger = LoggerUtil.getLogger();  
 @Resource  
 private ToDoItemsService toDoItemsService;  
 @Resource  
 private ItspServicesService servicesService;  
 @Resource  
 private ItspServicesApprovalService itspServicesApprovalService;  
 @Resource  
 private OrganService organService;  
 @Resource  
 private UserService userService;  
 @Resource  
 private ApprovalConfigService approvalConfigService;  
 @Resource  
 private ItspBMCJudgeConditionService itspBMCJudgeConditionService;  
 @Resource  
 private ItspServicesCatalogMappingService itspServicesCatalogMappingService;  
 @Resource  
 private IncidentServices incidentServices;  
 @Resource  
 private OperAuditService operAuditService;  
 @Resource  
 private WechatITSPUtil wechatITSPUtil;  
 @Resource  
 IncidentService incidentService;  
 @Resource  
 RequestApprovalSync requestApprovalSync;  
 @Resource  
 ItspNeedService itspNeedService;  
 @Resource  
 CntpVpNeed cntpVpNeed;  
 @Resource  
 PROBIncidentSync probIncidentSync;  
 @Resource  
 ToDoNum toDoNum;  
 @Resource  
 LoggerRecordService loggerRecordService;  
 @Override  
 public Map<String, Object> findApprovalByparam(PageParam<IToDoItemsVo> pageResult, IToDoItemsVo toDoItemsVo, String likeText,  
 ToDoTier[] searchTodoTiers, ToDoTier[] searchExceptTodoTiers) {  
 Map<String, Object> dataMap = new HashMap<String, Object>();  
 Date startDate = null;  
 Date endDate = new Date();  
 int completedCNT = 0;  
 int notCompletedCNT = 0;  
 int allCNT = 0;  
 //TODO 按照区间查询  
 if (null != toDoItemsVo.getDateParam()) {  
 if (DateParam.LAST\_ONE\_WEEK == toDoItemsVo.getDateParam()) {  
 startDate = DateUtils.AddDay(endDate, -7);  
 }  
 if (DateParam.LAST\_TWO\_WEEK == toDoItemsVo.getDateParam()) {  
 startDate = DateUtils.AddDay(endDate, -14);  
 }  
 if (DateParam.LAST\_THREE\_WEEK == toDoItemsVo.getDateParam()) {  
 startDate = DateUtils.AddDay(endDate, -21);  
 }  
 if (DateParam.LAST\_ONE\_MONTH == toDoItemsVo.getDateParam()) {  
 startDate = DateUtils.addMonth(endDate, -1);  
 }  
 if (DateParam.LAST\_TWO\_MONTH == toDoItemsVo.getDateParam()) {  
 startDate = DateUtils.addMonth(endDate, -2);  
 }  
 if (DateParam.LAST\_THREE\_MONTH == toDoItemsVo.getDateParam()) {  
 startDate = DateUtils.addMonth(endDate, -3);  
 }  
 if (DateParam.LAST\_HALF\_YEAR == toDoItemsVo.getDateParam()) {  
 startDate = DateUtils.addMonth(endDate, -6);  
 }  
 } else {  
 //默认当前日期前三个月记录  
 startDate = DateUtils.addMonth(endDate, -3);  
 //已完成  
 completedCNT = toDoItemsService.findItemsCount(toDoItemsVo.getUserName(), ToDoType.APPROVAL\_TYPE, "COMPLETED", startDate, endDate,  
 searchTodoTiers, searchExceptTodoTiers);  
 //未完成  
 notCompletedCNT = toDoItemsService.findItemsCount(toDoItemsVo.getUserName(), ToDoType.APPROVAL\_TYPE, "NOT\_COMPLETED", startDate, endDate,  
 searchTodoTiers, searchExceptTodoTiers);  
 //全部  
 allCNT = toDoItemsService.findItemsCount(toDoItemsVo.getUserName(), ToDoType.APPROVAL\_TYPE, "", startDate, endDate, searchTodoTiers,  
 searchExceptTodoTiers);  
 }  
 //返回结果  
 PageParam<IToDoItemsVo> pagerVo = new PageParam<IToDoItemsVo>();  
 //查询结果  
 toDoItemsVo.setToDoType(ToDoType.APPROVAL\_TYPE);  
 PageParam<ToDoItems> pager = toDoItemsService.findItemsByParams(pageResult, toDoItemsVo, startDate, endDate, likeText, searchTodoTiers,  
 searchExceptTodoTiers);  
 if (pager != null && pager.getPageList().size() > 0) {  
 //循环查询结果,存放至Vo展示  
 List<IToDoItemsVo> toDoItemsList = ToDoUtils.toDoitemsPoTOvo(pager.getPageList());  
 pagerVo.setPageList(toDoItemsList);  
 pagerVo.setPageNo(pager.getPageNo());  
 pagerVo.setPageSize(pager.getPageSize());  
 pagerVo.setTotalSize(pager.getTotalSize());  
 }  
 dataMap.put("pagerVo", pagerVo);  
 dataMap.put("completedCNT", completedCNT);  
 dataMap.put("notCompletedCNT", notCompletedCNT);  
 dataMap.put("allCNT", allCNT);  
 return dataMap;  
 }  
 /\*\*  
 \* 审批通过  
 \*/  
 @Override  
 public Map<String, String> submitApproval(IUserInfoVo userInfoVo, String id, String approvalText, ItspBMCJudgeConditionVo judgeConditionVo,  
 String toDoId) {  
 String flag = "0";  
 Map<String, String> rtnMap = new HashMap<String, String>();  
 rtnMap.put("code", "0");  
 ToDoItems toDoItem = toDoItemsService.getToDoItemsById(toDoId);  
 //判断是否是取消的  
 if (toDoItem != null && toDoItem.getToDoStatus() == ToDoStatus.NEW) {  
 //查询结果  
 ItspServices services = servicesService.getItspServicesById(id);  
 if (null != services) {  
 //set审批状态  
 services.setApprovalStatus(true);  
 //set审批内容  
 services.setApprovalText(approvalText);  
 //set审批人姓名  
 services.setApprovalUserDisplayName(userInfoVo.getDisplayName());  
 //set审批人  
 services.setApprovalUserName(userInfoVo.getUserName());  
 try {  
 if (services instanceof ItspIncident) {  
 flag = "1";  
 ItspIncident incident = (ItspIncident) services;  
 this.incidentSubmitApproval(incident, userInfoVo, toDoId, judgeConditionVo, toDoItem);  
 //发送微信通知  
 wechatITSPUtil.sendBMCNotice(services.getSubmitterUserName(), "审批通过",  
 "你报告的流水号为" + services.getBusNo() + "故障已经由" + userInfoVo.getUserName() + "审批通过，审批意见：" + approvalText + " 请知悉。",  
 SysConstant.MY\_APPRY\_INFO(services.getId()));  
 } else if (services instanceof ItspRequest) {  
 flag = "2";  
 ItspRequest itspRequest = (ItspRequest) services;  
 StatefulRtnBase requestSubmitApproval = this.requestSubmitApproval(itspRequest, userInfoVo, toDoItem);  
 if (null == requestSubmitApproval || requestSubmitApproval.getErrorCode() == "0") {  
 return rtnMap;  
 }  
 } else if (services instanceof ItspChange) {  
 flag = "3";  
 ItspChange itspchange = (ItspChange) services;  
 StatefulRtnBase requestSubmitApproval = this.changeSubmitApproval(itspchange, userInfoVo, toDoItem);  
 if (null == requestSubmitApproval || requestSubmitApproval.getErrorCode() == "0") {  
 return rtnMap;  
 }  
 } else if (services instanceof ItspRelease) {  
 flag = "4";  
 ItspRelease itspRelease = (ItspRelease) services;  
 StatefulRtnBase requestSubmitApproval = this.releaseSubmitApproval(itspRelease, userInfoVo, toDoItem);  
 if (null == requestSubmitApproval || requestSubmitApproval.getErrorCode() == "0") {  
 return rtnMap;  
 }  
 } else if (services instanceof ItspNeed) {  
 flag = "5";  
 ItspNeed itspRelease = (ItspNeed) services;  
 rtnMap = this.itspNeedSubmitApproval(itspRelease, userInfoVo, toDoItem);  
 if (null == rtnMap) {  
 return rtnMap;  
 }  
 } else {  
 throw new WechatBusinessException("不支持的服务类型信息");  
 }  
 } catch (Exception e) {  
 logger.error("提交审批出错:" + e);  
 if (flag == "1") {  
 loggerRecordService.saveLoggerRecord("ITSP", "ITSP", LoggerType.BUSINESS,ItspServicesType.INCIDENT ,   
 services.getBusNo(), null, "ITSP故障审批单", "ITSP故障审批单通过失败:" + e , null);  
 }else if(flag=="2"){  
 loggerRecordService.saveLoggerRecord("ITSP", "ITSP", LoggerType.BUSINESS,ItspServicesType.REQUEST ,   
 services.getBusNo(), null, "ITSP请求审批单", "ITSP请求审批单通过失败:" + e , null);  
 }else if(flag=="3"){  
 loggerRecordService.saveLoggerRecord("ITSP", "ITSP", LoggerType.BUSINESS,ItspServicesType.CHANGE ,   
 services.getBusNo(), null, "ITSP变更审批单", "ITSP变更审批单通过失败:" + e , null);  
 }else if(flag=="4"){  
 loggerRecordService.saveLoggerRecord("ITSP", "ITSP", LoggerType.BUSINESS,ItspServicesType.RELEASE ,   
 services.getBusNo(), null, "ITSP发布审批单", "ITSP发布审批单通过失败:" + e , null);  
 }else if(flag=="5"){  
 loggerRecordService.saveLoggerRecord("ITSP", "ITSP", LoggerType.BUSINESS,ItspServicesType.NEED ,   
 services.getBusNo(), null, "ITSP需求审批单", "ITSP需求审批单通过失败:" + e, null);  
 }  
 return rtnMap;  
 }  
 rtnMap.put("code", "1");  
 }  
 }  
  
 return rtnMap;  
 }  
 /\*\*  
 \* 审批驳回  
 \*/  
 @Override  
 public Map<String, String> rejectApproval(IUserInfoVo userInfoVo, ItspServicesVo itspServicesVo, String toDoId) {  
 Map<String, String> rtnMap = new HashMap<String, String>();  
 rtnMap.put("code", "0");  
 if (itspServicesVo == null || itspServicesVo.getId() == null) {  
 return rtnMap;  
 }  
 ToDoItems toDoItem = toDoItemsService.getToDoItemsById(toDoId);  
 //判断是否是取消的  
 if (toDoItem != null && toDoItem.getToDoStatus() == ToDoStatus.NEW) {  
 //查询结果  
 ItspServices itspServices = servicesService.getItspServicesById(itspServicesVo.getId());  
 if (null != itspServices) {  
 //set审批状态  
 itspServices.setApprovalStatus(true);  
 //set审批人姓名  
 itspServices.setApprovalUserDisplayName(userInfoVo.getDisplayName());  
 //set审批人  
 itspServices.setApprovalUserName(userInfoVo.getUserName());  
 //set审批内容  
 itspServices.setApprovalText(itspServicesVo.getApprovalText());  
 //set最后修改时间  
 itspServices.setLastUpdateDttm(new Date());  
 //set最后修改人  
 itspServices.setLastUpdateUser(userInfoVo.getUserName());  
 //set最后修改人姓名  
 itspServices.setLastUpdateUserName(userInfoVo.getDisplayName());  
 String wxMessage = "";  
 boolean sendWxMsg = true; // 发送微信消息标志位  
 if (itspServices instanceof ItspIncident) {  
 ItspIncident incident = (ItspIncident) itspServices;  
 this.incidentRejectApproval(incident, userInfoVo);  
 wxMessage = "你报告的流水号为" + itspServices.getBusNo() + "故障已经由" + userInfoVo.getUserName() + "审批驳回，审批意见："  
 + itspServicesVo.getApprovalText() + "。请知悉。如需修改信息后重新提交，请直接添加故障工作信息说明情况，工作信息发表后即提交成功！";  
 } else if (itspServices instanceof ItspRequest) {  
 ItspRequest itspRequest = (ItspRequest) itspServices;  
 StatefulRtnBase statefulRtnBase = this.requestRejectApproval(itspRequest, userInfoVo, toDoItem);  
 if (statefulRtnBase.isReqRtnStatus()) {  
 wxMessage = "你创建的流水号为" + itspServices.getBusNo() + "服务请求已经由" + userInfoVo.getUserName() + "审批驳回，审批意见："  
 + itspServicesVo.getApprovalText() + "。请知悉。";  
 } else {  
 return rtnMap;  
 }  
 } else if (itspServices instanceof ItspChange) {  
 ItspChange itspChange = (ItspChange) itspServices;  
 StatefulRtnBase statefulRtnBase = this.changeRejectApproval(itspChange, userInfoVo, toDoItem);  
 if (statefulRtnBase.isReqRtnStatus()) {  
 wxMessage = "你创建的流水号为" + itspServices.getBusNo() + "变更审批单已经由" + userInfoVo.getUserName() + "审批驳回，审批意见："  
 + itspServicesVo.getApprovalText() + "。请知悉。";  
 } else {  
 return rtnMap;  
 }  
 } else if (itspServices instanceof ItspRelease) {  
 ItspRelease itspRelease = (ItspRelease) itspServices;  
 StatefulRtnBase statefulRtnBase = this.releaseRejectApproval(itspRelease, userInfoVo, toDoItem);  
 if (statefulRtnBase.isReqRtnStatus()) {  
 wxMessage = "你创建的流水号为" + itspServices.getBusNo() + "发布审批单已经由" + userInfoVo.getUserName() + "审批驳回，审批意见："  
 + itspServicesVo.getApprovalText() + "。请知悉。";  
 } else {  
 return rtnMap;  
 }  
 } else if (itspServices instanceof ItspNeed) {  
 sendWxMsg = false;  
 ItspNeed itspRelease = (ItspNeed) itspServices;  
 rtnMap = this.itspNeedRejectApproval(itspRelease, userInfoVo, toDoItem);  
 if (null == rtnMap) {  
 return rtnMap;  
 }  
 } else {  
 throw new WechatBusinessException("不支持的服务类型信息");  
 }  
 if (sendWxMsg) {  
 //发送微信通知  
 wechatITSPUtil.sendBMCNotice(itspServices.getSubmitterUserName(), "审批驳回", wxMessage,  
 SysConstant.MY\_APPRY\_INFO(itspServices.getId()));  
 }  
 rtnMap.put("code", "1");  
 }  
 }  
 return rtnMap;  
 }  
 @Override  
 public List<ItspBMCJudgeConditionVo> findIsLastApproavl(ItspIncidentVo incidentVo) {  
 // 人员机构的包含关系  
 User user = userService.getUserByName(incidentVo.getSubmitterUserName());  
 List<String> organs = userService.getUserAllLevelOrganIds(user);  
  
 //查询审批规则  
 ItspServicesApprovalConfig itspServicesApprovalConfig = approvalConfigService  
 .getIncidentReporterCateOrganApprovalConfig(incidentVo.getCategory1(), incidentVo.getCategory2(), organs);  
  
 //根据当前审批级别获取下一级的审批规则  
 UserGroup userGroup = null;  
 //判断当前审批级别是否存在,如果为null,默认为1级审批  
 if (incidentVo.getApprovalLevel() == null) {  
 incidentVo.setApprovalLevel(1);  
 } else {  
 incidentVo.setApprovalLevel(incidentVo.getApprovalLevel() + 1);  
 }  
 if (incidentVo.getApprovalLevel() == 1) {  
 userGroup = itspServicesApprovalConfig.getApprovalGroup2();  
 } else if (incidentVo.getApprovalLevel() == 2) {  
 userGroup = itspServicesApprovalConfig.getApprovalGroup3();  
 }  
 List<ItspBMCJudgeConditionVo> judgeConditionVoList = new ArrayList<ItspBMCJudgeConditionVo>();  
 if (userGroup == null) {  
 // 得到映射的判断维度  
 ItspServicesCatalogMapping catalogMapping = itspServicesCatalogMappingService.getItspToBMCCatalogCode(incidentVo.getCategory1(),  
 incidentVo.getCategory2(), organs, user.getWorkPlaceClass());  
 if (catalogMapping == null) {  
 throw new WechatBusinessException("您报障的分类数据出现问题，请联系管理员!");  
 }  
 List<ItspBMCJudgeCondition> judgeConditionList = itspBMCJudgeConditionService.findBMCJudgeConditionByJudgeCondition(catalogMapping);  
 try {  
 for (ItspBMCJudgeCondition judgeCondition : judgeConditionList) {  
 ItspBMCJudgeConditionVo judgeConditionVo = new ItspBMCJudgeConditionVo();  
 BeanUtils.copyProperties(judgeConditionVo, judgeCondition);  
 judgeConditionVoList.add(judgeConditionVo);  
 }  
 } catch (IllegalAccessException e) {  
 logger.error("属性的IllegalAccessException问题", e);  
 } catch (InvocationTargetException e) {  
 logger.error("属性的InvocationTargetException问题", e);  
 }  
 }  
 return judgeConditionVoList;  
 }  
 @Override  
 public Map<String, Object> findApprovalGroupPeople(String userId, String category1, String category2, Integer approvleLevel) {  
 Map<String, Object> rtnMap = new HashMap<String, Object>();  
 //先查询出人员的相关信息  
 User user = userService.getUserById(userId);  
 // 人员机构的包含关系  
 List<String> organs = userService.getUserAllLevelOrganIds(user);  
 //查询审批规则  
 ItspServicesApprovalConfig itspServicesApprovalConfig = approvalConfigService.getIncidentReporterCateOrganApprovalConfig(category1, category2,  
 organs);  
 List<IUserInfoVo> userInfoList = new ArrayList<IUserInfoVo>();  
 if (itspServicesApprovalConfig != null && itspServicesApprovalConfig.getApprovalLevel() > 0) {  
 //根据当前审批级别获取下一级的审批规则  
 UserGroup userGroup = null;  
 if (approvleLevel == null || approvleLevel == 0) {  
 userGroup = itspServicesApprovalConfig.getApprovalGroup1();  
 } else if (approvleLevel == 1) {  
 userGroup = itspServicesApprovalConfig.getApprovalGroup2();  
 } else if (approvleLevel == 2) {  
 userGroup = itspServicesApprovalConfig.getApprovalGroup3();  
 }  
 //如果不等于null,则证明有下一级审批  
 if (userGroup != null) {  
 // 审批人员  
 List<UserGroupMembers> memberList = userGroup.getUserGroupMembersList();  
 if (memberList != null && memberList.size() > 0) {  
 for (UserGroupMembers member : memberList) {  
 User u = userService.getUserByName(member.getUserOrganId());  
 if (null != u) {  
 IUserInfoVo userInfoVo = new IUserInfoVo();  
 userInfoVo.setDisplayName(u.getDisplayName());  
 userInfoVo.setUserName(u.getUserName());  
 userInfoList.add(userInfoVo);  
 }  
 }  
 }  
 }  
 rtnMap.put("userInfoList", userInfoList);  
 rtnMap.put("userGroupName", (userGroup == null) ? "" : userGroup.getName());  
 }  
 return rtnMap;  
 }  
 public void incidentSubmitApproval(ItspIncident incident, IUserInfoVo userInfoVo, String toDoId, ItspBMCJudgeConditionVo judgeConditionVo,  
 ToDoItems toDoItem) {  
  
 //判断当前审批级别是否存在,如果为null,默认为1级审批  
 if (incident.getApprovalLevel() == null) {  
 incident.setApprovalLevel(1);  
 } else {//如果不为null,则在原本的审批级别上+1  
 incident.setApprovalLevel(incident.getApprovalLevel() + 1);  
 }  
 //查询审批组所有有关此故障单的待办  
 List<ToDoItems> toDoItemsList = toDoItemsService.findItemsByServicesId(incident.getId(), new ToDoType[] { ToDoType.APPROVAL\_TYPE },  
 new ToDoStatus[] { ToDoStatus.NEW, ToDoStatus.FINISH });  
 if (toDoItemsList != null && toDoItemsList.size() > 0) {  
 //修改审批组人员的待办状态(除审批提交人 )  
 for (ToDoItems toDo : toDoItemsList) {  
 if (toDo.getToDoStatus() == ToDoStatus.NEW) {  
 //如果待办的人员信息不等于审批提交人,进行修改  
 if (!toDo.getUserName().equals(userInfoVo.getUserName())) {  
 //状态修改为取消状态  
 toDo.setToDoStatus(ToDoStatus.CANCEL);  
 toDo.setLastUpdateDttm(new Date());  
 toDoItemsService.saveOrUpdateToDoItems(toDo);  
 toDoNum.findByUserToDoNum(toDo.getUserName());  
 } else {  
 toDo.setToDoStatus(ToDoStatus.FINISH);  
 toDo.setLastUpdateDttm(new Date());  
 toDoItemsService.saveOrUpdateToDoItems(toDo);  
 toDoNum.findByUserToDoNum(toDo.getUserName());  
 }  
 } else { // 历史完成的  
 if (toDo.getUserName().equals(userInfoVo.getUserName())) {  
 //状态修改为取消状态,合并为一个已完成审批  
 toDo.setToDoStatus(ToDoStatus.CANCEL);  
 toDo.setLastUpdateDttm(new Date());  
 toDoItemsService.saveOrUpdateToDoItems(toDo);  
 toDoNum.findByUserToDoNum(toDo.getUserName());  
 }  
 }  
 }  
 }  
 //查询是否还有下一级审批:  
 //先查询出人员的相关信息  
 User user = userService.getUserById(userInfoVo.getId());  
 // 人员机构的包含关系  
 List<String> organs = userService.getUserAllLevelOrganIds(user);  
 //查询审批规则  
 ItspServicesApprovalConfig itspServicesApprovalConfig = approvalConfigService  
 .getIncidentReporterCateOrganApprovalConfig(incident.getCategory1(), incident.getCategory2(), organs);  
 if (itspServicesApprovalConfig != null && itspServicesApprovalConfig.getApprovalLevel() > 0) {  
 //根据当前审批级别获取下一级的审批规则  
 UserGroup userGroup = null;  
 if (incident.getApprovalLevel() == 1) {  
 userGroup = itspServicesApprovalConfig.getApprovalGroup2();  
 } else if (incident.getApprovalLevel() == 2) {  
 userGroup = itspServicesApprovalConfig.getApprovalGroup3();  
 }  
 //如果不等于null,则证明有下一级审批  
 if (userGroup != null) {  
 // 审批人员  
 List<UserGroupMembers> memberList = userGroup.getUserGroupMembersList();  
 if (memberList != null && memberList.size() > 0) {  
 for (UserGroupMembers member : memberList) {  
 // 新建组下的人员待办审批任务  
 ToDoItems toDoItems = new ToDoItems();  
 toDoItems.setItspServices(incident);  
 toDoItems.setToDoStatus(ToDoStatus.NEW);  
 toDoItems.setToDoTier(ToDoTier.INCIDENT\_TIER);  
 toDoItems.setCrtUser(member.getUserOrganId());  
 toDoItems.setLastUpdateUser(member.getUserOrganName());  
 toDoItems.setToDoType(ToDoType.APPROVAL\_TYPE);  
 toDoItems.setUserName(member.getUserOrganId());  
 toDoItemsService.saveOrUpdateToDoItems(toDoItems);  
 toDoNum.findByUserToDoNum(toDoItems.getUserName());  
 wechatITSPUtil.sendBMCNotice(member.getUserOrganName(), "故障待审批",  
 "有一则故障报告待您审批，故障报告流水号为" + incident.getBusNo() + ",请尽快登录IT服务门户审批，谢谢！",  
 SysConstant.MY\_APPROVAL\_INFO(incident.getId(), toDoItems.getId()));  
 }  
 }  
 } else { //如果等于null,则证明没有下一级审批  
 //没有的下一级审批的话 ,状态置为submit  
 incident.setPlatStatus(ItspServiceStatus.Submit);  
 incident.setSubmitType(SubmitType.WAIT);  
 incident.setApprovalFinishStatusTime(new Date());  
 incident.setCurrentUserName(user.getUserName());  
 incident.setCurrentUserDisplayName(user.getDisplayName());  
 incident.setInfluenceRange(judgeConditionVo.getImpactScope()); // 影响范围   
 incident.setInfluenceChannel(judgeConditionVo.getImpactChannel()); // 影响渠道   
 incident.setInfluenceFunction(judgeConditionVo.getImpactFunc()); // 影响功能   
 incident.setFaultScene(judgeConditionVo.getIncidentScene()); // 故障场景   
 incident.setLastUpdateUser("ITSP");  
 if (StringUtils.isBlank(judgeConditionVo.getImpact())) {  
 //查询出影响度  
 try {  
 ItspBMCJudgeCondition judgeCondition = new ItspBMCJudgeCondition();  
 BeanUtils.copyProperties(judgeCondition, judgeConditionVo);  
 judgeCondition = itspBMCJudgeConditionService.getBMCJudgeCondition(judgeCondition);  
 incident.setSourceInfluence(judgeCondition.getImpact());  
 } catch (IllegalAccessException e) {  
 logger.error("属性的IllegalAccessException问题", e);  
 } catch (InvocationTargetException e) {  
 logger.error("属性的InvocationTargetException问题", e);  
 }  
 } else {  
 incident.setSourceInfluence(judgeConditionVo.getImpact());  
 }  
 incidentService.saveOrUpdate(incident);  
  
 //提交故障单  
 incidentServices.submitIncident(incident.getId(), userInfoVo);  
 }  
 //添加审计日志  
 ItspServicesOperAudit operAudit = new ItspServicesOperAudit();  
 operAudit.setItspServices(incident);  
 operAudit.setiTSPServiceSourceSys(ItspServiceSourceSys.ITSP);  
 operAudit.setCrtDttm(new Date());  
 operAudit.setCrtUser(user.getUserName());  
 operAudit.setCrtUserName(user.getUserName());  
 operAudit.setLastUpdateDttm(new Date());  
 operAudit.setLastUpdateUser(user.getUserName());  
 operAudit.setLastUpdateUserName(user.getUserName());  
 operAudit.setOperContent(incident.getApprovalText());  
 operAudit.setOperDate(new Date());  
 operAudit.setOperType("新建");  
 operAudit.setOperUserDisplayName(user.getDisplayName());  
 operAudit.setOperUserId(user.getId());  
 operAudit.setOperUserName(user.getUserName());  
 if (incident.getIsRecorder()) {  
 operAudit.setSyncStatus(SyncStatus.WAIT);  
 operAudit.setSyncReSendNo(0);  
 }  
 operAuditService.saveOrUpdate(operAudit);  
 if (incident.getIsRecorder()) {  
 probIncidentSync.submitIncidentAudit(operAudit.getId());  
 }  
 //修改待办状态  
 if (StringUtils.isNotBlank(toDoId)) {  
 toDoItem = toDoItemsService.getToDoItemsById(toDoId);  
 toDoItem.setToDoStatus(ToDoStatus.FINISH);  
 toDoItem.setLastUpdateDttm(new Date());  
 toDoItemsService.saveOrUpdateToDoItems(toDoItem);  
 toDoNum.findByUserToDoNum(toDoItem.getUserName());  
 }  
  
 }  
 }  
 public void incidentRejectApproval(ItspIncident incident, IUserInfoVo userInfoVo)   
 //驳回:状态置为Return  
 incident.setPlatStatus(ItspServiceStatus.Return);  
 //set当前审批级别  
 incident.setApprovalLevel(0);  
 //修改故障审批信息  
 servicesService.updateServices(incident);  
 //添加审计日志  
 ItspServicesOperAudit operAudit = new ItspServicesOperAudit();  
 operAudit.setItspServices(incident);  
 operAudit.setiTSPServiceSourceSys(ItspServiceSourceSys.ITSP);  
 operAudit.setCrtDttm(new Date());  
 operAudit.setCrtUser(userInfoVo.getUserName());  
 operAudit.setCrtUserName(userInfoVo.getUserName());  
 operAudit.setLastUpdateDttm(new Date());  
 operAudit.setLastUpdateUser(userInfoVo.getUserName());  
 operAudit.setLastUpdateUserName(userInfoVo.getUserName());  
 operAudit.setOperContent("审批驳回，原因：" + incident.getApprovalText());  
 operAudit.setOperDate(new Date());  
 operAudit.setOperType("新建");  
 operAudit.setOperUserDisplayName(userInfoVo.getDisplayName());  
 operAudit.setOperUserId(userInfoVo.getId());  
 operAudit.setOperUserName(userInfoVo.getUserName());  
 if (incident.getIsRecorder()) {  
 operAudit.setSyncStatus(SyncStatus.WAIT);  
 operAudit.setSyncReSendNo(0);  
 }  
 operAuditService.saveOrUpdate(operAudit);  
 if (incident.getIsRecorder()) {  
 probIncidentSync.submitIncidentAudit(operAudit.getId());  
 }  
 //查询审批组所有有关此故障单的待办  
 List<ToDoItems> toDoItemsList = toDoItemsService.findItemsByServicesId(incident);  
 if (toDoItemsList != null && toDoItemsList.size() > 0) {  
 //修改审批组人员的待办状态  
 for (ToDoItems toDo : toDoItemsList) {  
 //状态修改为取消状态  
 //如果待办的人员信息不等于审批提交人,进行修改  
 if (!toDo.getUserName().equals(userInfoVo.getUserName())) {  
 // 非本人的状态修改为取消状态  
 toDo.setToDoStatus(ToDoStatus.CANCEL);  
 toDo.setLastUpdateDttm(new Date());  
 toDoItemsService.saveOrUpdateToDoItems(toDo);  
 toDoNum.findByUserToDoNum(toDo.getUserName());  
 } else {  
 toDo.setToDoStatus(ToDoStatus.FINISH);  
 toDo.setLastUpdateDttm(new Date());  
 toDoItemsService.saveOrUpdateToDoItems(toDo);  
 toDoNum.findByUserToDoNum(toDo.getUserName());  
 }  
 }  
 }  
 }  
 public StatefulRtnBase requestSubmitApproval(ItspRequest itspRequest, IUserInfoVo userInfoVo, ToDoItems toDoItems) {  
 //先查询出人员的相关信息  
 User user = userService.getUserById(userInfoVo.getId());  
 StatefulRtnBase rtnResult = new StatefulRtnBase();  
 if (null != toDoItems) {  
 //审批记录   
 ItspServicesApproval itspServicesApproval = new ItspServicesApproval();  
 itspServicesApproval.setItspServices(itspRequest);  
 itspServicesApproval.setApprovalStatus(ItspApprovalStatus.Approved);// 审批状态 必填/字符/254   
 itspServicesApproval.setSigTermStateDate(new Date());// 签名日期 必填/字符/254  
 itspServicesApproval.setApprovalText(itspRequest.getApprovalText());// 审批意见 审批意见/审批结果 (通过说明或者退回意见) 非必填/字符/254   
 itspServicesApproval.setSignatureId(toDoItems.getSignatureID()); // 审批ID 必填/字符/254   
 itspServicesApproval.setApprovalUserId(user.getUserName()); // 审批人ID 必填/字符/254   
 itspServicesApproval.setApprovalUserName(user.getUserName()); // 审批人签名UserName 必填/字符/254   
 itspServicesApproval.setApprovalUserDisplayName(user.getDisplayName()); // 审批人姓名   
 itspServicesApproval.setApprovalEmail(user.getEmail()); // 审批人邮箱 必填/字符/254   
 itspServicesApproval.setCrtDttm(new Date()); //创建时间 createDate 必填/数值型 时间戳　   
 itspServicesApproval.setLastUpdateDttm(new Date()); //修改时间 lastUpdateTime 必填/数值型 　时间戳   
 itspServicesApproval.setLastUpdateUser("ITSP");  
 //提交审批记录  
 rtnResult = requestApprovalSync.submitRequestApproval(itspServicesApproval, itspRequest, toDoItems, userInfoVo);  
 if (null != rtnResult && rtnResult.isReqRtnStatus()) {  
 //记录到本地  
 //itspServicesApprovalService.saveOrUpdate(itspServicesApproval);  
 //修改待办状态  
 /\* toDoItems.setToDoStatus(ToDoStatus.FINISH);  
 toDoItems.setLastUpdateDttm(new Date());  
 toDoItemsService.saveOrUpdateToDoItems(toDoItems);\*/  
 }  
 }  
 return rtnResult;  
 }  
 /\*\*  
 \* 服务请求审批驳回  
 \* 周健鹏  
 \* 2016年10月25日下午3:19:09  
 \*/  
 public StatefulRtnBase requestRejectApproval(ItspRequest itspRequest, IUserInfoVo userInfoVo, ToDoItems toDoItems) {  
 //驳回:状态置为Draft  
 //itspRequest.setPlatStatus(ItspServiceStatus.Draft);  
 //修改故障审批信息  
 //servicesService.updateServices(itspRequest);  
 //查询审批待办  
 List<ToDoItems> toDoItemsList = toDoItemsService.findItemsByServicesId(itspRequest.getId(), new ToDoType[] { ToDoType.APPROVAL\_TYPE },  
 new ToDoStatus[] { ToDoStatus.NEW, ToDoStatus.FINISH });  
 if (toDoItemsList != null && toDoItemsList.size() > 0) {  
 //修改审批组人员的待办状态  
 for (ToDoItems toDo : toDoItemsList) {  
 //状态修改为取消状态  
 //如果待办的人员信息不等于审批提交人,进行修改  
 if (!toDo.getUserName().equals(userInfoVo.getUserName())) {  
 // 非本人的状态修改为取消状态  
 toDo.setToDoStatus(ToDoStatus.CANCEL);  
 toDo.setLastUpdateDttm(new Date());  
 toDoItemsService.saveOrUpdateToDoItems(toDo);  
 toDoNum.findByUserToDoNum(toDo.getUserName());  
 } else {  
 toDo.setToDoStatus(ToDoStatus.FINISH);  
 toDo.setLastUpdateDttm(new Date());  
 toDoItemsService.saveOrUpdateToDoItems(toDo);  
 toDoNum.findByUserToDoNum(toDo.getUserName());  
 }  
 }  
 }  
 //先查询出人员的相关信息  
 Use user = userService.getUserById(userInfoVo.getId());  
 //审批记录   
 ItspServicesApproval itspServicesApproval = new ItspServicesApproval();  
 itspServicesApproval.setItspServices(itspRequest);  
 itspServicesApproval.setApprovalStatus(ItspApprovalStatus.Rejected);// 审批状态 必填/字符/254   
 itspServicesApproval.setSigTermStateDate(new Date());// 签名日期 必填/字符/254  
 itspServicesApproval.setApprovalText(itspRequest.getApprovalText());// 审批意见 审批意见/审批结果 (通过说明或者退回意见) 非必填/字符/254   
 //itspServicesApproval.setSignatureId(itspRequest.getSignatureID()); // 审批ID 必填/字符/254   
 itspServicesApproval.setApprovalUserId(user.getUserName()); // 审批人ID 必填/字符/254   
 itspServicesApproval.setApprovalUserName(user.getUserName()); // 审批人签名UserName 必填/字符/254   
 itspServicesApproval.setApprovalUserDisplayName(user.getDisplayName()); // 审批人姓名   
 itspServicesApproval.setApprovalEmail(user.getEmail()); // 审批人邮箱 必填/字符/254   
 itspServicesApproval.setCrtDttm(new Date()); //创建时间 createDate 必填/数值型 时间戳　   
 itspServicesApproval.setLastUpdateDttm(new Date()); //修改时间 lastUpdateTime 必填/数值型 　时间戳   
 itspServicesApproval.setLastUpdateUser("ITSP");  
 //itspServicesApprovalService.saveOrUpdate(itspServicesApproval);  
 //提交审批记录  
 StatefulRtnBase statefulRtnBase = requestApprovalSync.submitRequestApproval(itspServicesApproval, itspRequest, toDoItems, userInfoVo);  
 /\* //修改待办状态  
 toDoItems.setToDoStatus(ToDoStatus.FINISH);  
 toDoItems.setLastUpdateDttm(new Date());  
 toDoItemsService.saveOrUpdateToDoItems(toDoItems);\*/  
 return statefulRtnBase;  
 }  
 /\*\*  
 \*变更审批通过  
 \* 陈恒  
 \* 2016年12月9日上午10:10:39  
 \*/  
 public StatefulRtnBase changeSubmitApproval(ItspChange itspChenge, IUserInfoVo userInfoVo, ToDoItems toDoItems) {  
 //先查询出人员的相关信息  
 User user = userService.getUserById(userInfoVo.getId());  
 StatefulRtnBase rtnResult = new StatefulRtnBase();  
 if (null != toDoItems) {  
 //审批记录   
 ItspServicesApproval itspServicesApproval = new ItspServicesApproval();  
 itspServicesApproval.setItspServices(itspChenge);  
 itspServicesApproval.setApprovalStatus(ItspApprovalStatus.Approved);// 审批状态 必填/字符/254   
 itspServicesApproval.setSigTermStateDate(new Date());// 签名日期 必填/字符/254  
 itspServicesApproval.setApprovalText(itspChenge.getApprovalText());// 审批意见 审批意见/审批结果 (通过说明或者退回意见) 非必填/字符/254   
 itspServicesApproval.setSignatureId(toDoItems.getSignatureID()); // 审批ID 必填/字符/254   
 itspServicesApproval.setApprovalUserId(user.getUserName()); // 审批人ID 必填/字符/254   
 itspServicesApproval.setApprovalUserName(user.getUserName()); // 审批人签名UserName 必填/字符/254   
 itspServicesApproval.setApprovalUserDisplayName(user.getDisplayName()); // 审批人姓名   
 itspServicesApproval.setApprovalEmail(user.getEmail()); // 审批人邮箱 必填/字符/254   
 itspServicesApproval.setCrtDttm(new Date()); //创建时间 createDate 必填/数值型 时间戳　   
 itspServicesApproval.setLastUpdateDttm(new Date()); //修改时间 lastUpdateTime 必填/数值型 　时间戳   
 itspServicesApproval.setLastUpdateUser("ITSP");  
 //提交审批记录  
 rtnResult = requestApprovalSync.submitRequestApproval(itspServicesApproval, itspChenge, toDoItems, userInfoVo);  
 }  
 return rtnResult;  
 }  
 /\*\*  
 \*变更审批拒绝  
 \* 陈恒  
 \* 2016年12月9日上午10:10:39  
 \*/  
 public StatefulRtnBase changeRejectApproval(ItspChange itspChenge, IUserInfoVo userInfoVo, ToDoItems toDoItems) {  
 //先查询出人员的相关信息  
 User user = userService.getUserById(userInfoVo.getId());  
 //审批记录   
 ItspServicesApproval itspServicesApproval = new ItspServicesApproval();  
 itspServicesApproval.setItspServices(itspChenge);  
 itspServicesApproval.setApprovalStatus(ItspApprovalStatus.Rejected);// 审批状态 必填/字符/254   
 itspServicesApproval.setSigTermStateDate(new Date());// 签名日期 必填/字符/254  
 itspServicesApproval.setApprovalText(itspChenge.getApprovalText());// 审批意见 审批意见/审批结果 (通过说明或者退回意见) 非必填/字符/254   
 //itspServicesApproval.setSignatureId(itspRequest.getSignatureID()); // 审批ID 必填/字符/254   
 itspServicesApproval.setApprovalUserId(user.getUserName()); // 审批人ID 必填/字符/254   
 itspServicesApproval.setApprovalUserName(user.getUserName()); // 审批人签名UserName 必填/字符/254   
 itspServicesApproval.setApprovalUserDisplayName(user.getDisplayName()); // 审批人姓名   
 itspServicesApproval.setApprovalEmail(user.getEmail()); // 审批人邮箱 必填/字符/254   
 itspServicesApproval.setCrtDttm(new Date()); //创建时间 createDate 必填/数值型 时间戳　   
 itspServicesApproval.setLastUpdateDttm(new Date()); //修改时间 lastUpdateTime 必填/数值型 　时间戳   
 itspServicesApproval.setLastUpdateUser("ITSP");  
 //提交审批记录  
 StatefulRtnBase statefulRtnBase = requestApprovalSync.submitRequestApproval(itspServicesApproval, itspChenge, toDoItems, userInfoVo);  
 return statefulRtnBase;  
 }  
 /\*\*  
 \*发布审批通过  
 \* 陈恒  
 \* 2016年12月9日上午14:10:39  
 \*/  
 public StatefulRtnBase releaseSubmitApproval(ItspRelease itspRelease, IUserInfoVo userInfoVo, ToDoItems toDoItems) {  
 //先查询出人员的相关信息  
 User user = userService.getUserById(userInfoVo.getId());  
 StatefulRtnBase rtnResult = new StatefulRtnBase();  
 if (null != toDoItems) {  
 //审批记录   
 ItspServicesApproval itspServicesApproval = new ItspServicesApproval();  
 itspServicesApproval.setItspServices(itspRelease);  
 itspServicesApproval.setApprovalStatus(ItspApprovalStatus.Approved);// 审批状态 必填/字符/254   
 itspServicesApproval.setSigTermStateDate(new Date());// 签名日期 必填/字符/254  
 itspServicesApproval.setApprovalText(itspRelease.getApprovalText());// 审批意见 审批意见/审批结果 (通过说明或者退回意见) 非必填/字符/254   
 itspServicesApproval.setSignatureId(toDoItems.getSignatureID()); // 审批ID 必填/字符/254   
 itspServicesApproval.setApprovalUserId(user.getUserName()); // 审批人ID 必填/字符/254   
 itspServicesApproval.setApprovalUserName(user.getUserName()); // 审批人签名UserName 必填/字符/254   
 itspServicesApproval.setApprovalUserDisplayName(user.getDisplayName()); // 审批人姓名   
 itspServicesApproval.setApprovalEmail(user.getEmail()); // 审批人邮箱 必填/字符/254   
 itspServicesApproval.setCrtDttm(new Date()); //创建时间 createDate 必填/数值型 时间戳　   
 itspServicesApproval.setLastUpdateDttm(new Date()); //修改时间 lastUpdateTime 必填/数值型 　时间戳   
 itspServicesApproval.setLastUpdateUser("ITSP");  
 //提交审批记录  
 rtnResult = requestApprovalSync.submitRequestApproval(itspServicesApproval, itspRelease, toDoItems, userInfoVo);  
 }  
 return rtnResult;  
 }  
 /\*\*  
 \*发布审批拒绝  
 \* 陈恒  
 \* 2016年12月12日上午14:10:39  
 \*/  
 public StatefulRtnBase releaseRejectApproval(ItspRelease itspRelease, IUserInfoVo userInfoVo, ToDoItems toDoItems) {  
 //先查询出人员的相关信息  
 User user = userService.getUserById(userInfoVo.getId());  
 //审批记录   
 ItspServicesApproval itspServicesApproval = new ItspServicesApproval();  
 itspServicesApproval.setItspServices(itspRelease);  
 itspServicesApproval.setApprovalStatus(ItspApprovalStatus.Rejected);// 审批状态 必填/字符/254   
 itspServicesApproval.setSigTermStateDate(new Date());// 签名日期 必填/字符/254  
 itspServicesApproval.setApprovalText(itspRelease.getApprovalText());// 审批意见 审批意见/审批结果 (通过说明或者退回意见) 非必填/字符/254   
 //itspServicesApproval.setSignatureId(itspRequest.getSignatureID()); // 审批ID 必填/字符/254   
 itspServicesApproval.setApprovalUserId(user.getUserName()); // 审批人ID 必填/字符/254   
 itspServicesApproval.setApprovalUserName(user.getUserName()); // 审批人签名UserName 必填/字符/254   
 itspServicesApproval.setApprovalUserDisplayName(user.getDisplayName()); // 审批人姓名   
 itspServicesApproval.setApprovalEmail(user.getEmail()); // 审批人邮箱 必填/字符/254   
 itspServicesApproval.setCrtDttm(new Date()); //创建时间 createDate 必填/数值型 时间戳　   
 itspServicesApproval.setLastUpdateDttm(new Date()); //修改时间 lastUpdateTime 必填/数值型 　时间戳   
 itspServicesApproval.setLastUpdateUser("ITSP");  
 //提交审批记录  
 StatefulRtnBase statefulRtnBase = requestApprovalSync.submitRequestApproval(itspServicesApproval, itspRelease, toDoItems, userInfoVo);  
 return statefulRtnBase;  
 }  
 /\*\*  
 \* @param itspNeed  
 \* @param userInfoVo  
 \* @param toDoItems  
 \* @return  
 \*/  
 private Map<String, String> itspNeedSubmitApproval(ItspNeed itspNeed, IUserInfoVo userInfoVo, ToDoItems toDoItems) {  
 Map<String, String> rtnMap = new HashMap<String, String>();  
 rtnMap.put("code", "0");  
 if (null != toDoItems) {  
 if (null != itspNeed.getAppList()) {  
 //判断当前审批级别是否存在,如果为null,默认为1级审批  
 itspNeed.setApprovalLevel(1);  
 // 拆分子主需求  
 List<ItspNeed> itspNeedBranchList = itspNeedService.splitSaveNeed(itspNeed, userInfoVo);  
 //查询审批组所有有关此VP需求的待办  
 List<ToDoItems> toDoItemsList = toDoItemsService.findItemsByServicesId(itspNeed.getId(), new ToDoType[] { ToDoType.APPROVAL\_TYPE },  
 new ToDoStatus[] { ToDoStatus.NEW, ToDoStatus.FINISH });  
 if (toDoItemsList != null && toDoItemsList.size() > 0) {  
 //修改审批组人员的待办状态(除审批提交人 )  
 for (ToDoItems toDo : toDoItemsList) {  
 if (toDo.getToDoStatus() == ToDoStatus.NEW) {  
 if (!toDo.getUserName().equals(userInfoVo.getUserName())) {  
 //状态修改为取消状态  
 toDo.setToDoStatus(ToDoStatus.CANCEL);  
 toDo.setLastUpdateDttm(new Date());  
 toDoItemsService.saveOrUpdateToDoItems(toDo);  
 } else {  
 toDo.setToDoStatus(ToDoStatus.FINISH);  
 toDo.setLastUpdateDttm(new Date());  
 toDoItemsService.saveOrUpdateToDoItems(toDo);  
 }  
 } else { // 历史完成的  
 if (toDo.getUserName().equals(userInfoVo.getUserName())) {  
 //状态修改为取消状态,合并为一个已完成审批  
 toDo.setToDoStatus(ToDoStatus.CANCEL);  
 toDo.setLastUpdateDttm(new Date());  
 toDoItemsService.saveOrUpdateToDoItems(toDo);  
 }  
 }  
 }  
 }  
 // 将状态置为处理中  
 itspNeed.setPlatStatus(ItspServiceStatus.In\_Progress);  
 itspNeed.setCurrentUserName(userInfoVo.getUserName());  
 itspNeed.setCurrentUserDisplayName(userInfoVo.getDisplayName());  
 itspNeed.setLastUpdateUser("ITSP");  
 itspNeedService.saveOrUpdate(itspNeed);  
 for (ItspNeed itspNeedBranch : itspNeedBranchList) {  
 cntpVpNeed.asyncSubmitterItspNeed(itspNeedBranch.getBusNo(), userInfoVo);  
 }  
 rtnMap.put("code", "1");  
 }  
 }  
 return rtnMap;  
 }  
 /\*\*  
 \* @param itspNeed  
 \* @param userInfoVo  
 \* @param toDoItems  
 \* @return  
 \*/  
 private Map<String, String> itspNeedRejectApproval(ItspNeed itspNeed, IUserInfoVo userInfoVo, ToDoItems toDoItems) {  
 Map<String, String> rtnMap = new HashMap<String, String>();  
 rtnMap.put("code", "0");  
 if (null != toDoItems) {  
 //查询人员的相关信息  
 User user = userService.getUserById(userInfoVo.getId());  
 String operContent = "主需求由" + userInfoVo.getDisplayName() + "驳回";  
 // 添加审计日志  
 operAuditService.saveItspServicesOperAudit(user, itspNeed, operContent, "新建");  
 itspServicesApprovalService.saveItspServicesApproval(itspNeed, userInfoVo, ItspApprovalStatus.Return, null);  
 List<ToDoItems> toDoItemsList = toDoItemsService.findItemsByServicesId(itspNeed);  
 if (toDoItemsList != null && toDoItemsList.size() > 0) {  
 //修改审批组人员的待办状态  
 for (ToDoItems toDo : toDoItemsList) {  
 if (!toDo.getUserName().equals(userInfoVo.getUserName())) {  
 toDo.setToDoStatus(ToDoStatus.CANCEL);  
 toDo.setLastUpdateDttm(new Date());  
 toDoItemsService.saveOrUpdateToDoItems(toDo);  
 } else {  
 toDo.setToDoStatus(ToDoStatus.FINISH);  
 toDo.setLastUpdateDttm(new Date());  
 toDoItemsService.saveOrUpdateToDoItems(toDo);  
 }  
 }  
 }  
 // 将状态置为草稿  
 itspNeed.setPlatStatus(ItspServiceStatus.Draft);  
 itspNeed.setCurrentUserName(userInfoVo.getUserName());  
 itspNeed.setCurrentUserDisplayName(userInfoVo.getDisplayName());  
 itspNeed.setLastUpdateUser("ITSP");  
 itspNeedService.saveOrUpdate(itspNeed);  
 toDoItemsService.saveOrUpdateToDoItems(toDoItems);  
 rtnMap.put("code", "1");  
 }  
 return rtnMap;  
 }  
 @Override  
 public Map<String, Object> findApprovalByparamNew(PageParam<IToDoItemsVo> pageResult, IToDoItemsVo toDoItemsVo, String likeText,  
 ToDoTier[] searchTodoTiers, ToDoTier[] searchExceptTodoTiers) {  
 Map<String, Object> dataMap = new HashMap<String, Object>();  
 Date startDate = null;  
 Date endDate = new Date();  
 int completedCNT = 0;  
 int notCompletedCNT = 0;  
 int allCNT = 0;  
 //TODO 按照区间查询  
 if (null != toDoItemsVo.getDateParam()) {  
 if (DateParam.LAST\_ONE\_WEEK == toDoItemsVo.getDateParam()) {  
 startDate = DateUtils.AddDay(endDate, -7);  
 }  
 if (DateParam.LAST\_TWO\_WEEK == toDoItemsVo.getDateParam()) {  
 startDate = DateUtils.AddDay(endDate, -14);  
 }  
 if (DateParam.LAST\_THREE\_WEEK == toDoItemsVo.getDateParam()) {  
 startDate = DateUtils.AddDay(endDate, -21);  
 }  
 if (DateParam.LAST\_ONE\_MONTH == toDoItemsVo.getDateParam()) {  
 startDate = DateUtils.addMonth(endDate, -1);  
 }  
 if (DateParam.LAST\_TWO\_MONTH == toDoItemsVo.getDateParam()) {  
 startDate = DateUtils.addMonth(endDate, -2);  
 }  
 if (DateParam.LAST\_THREE\_MONTH == toDoItemsVo.getDateParam()) {  
 startDate = DateUtils.addMonth(endDate, -3);  
 }  
 if (DateParam.LAST\_HALF\_YEAR == toDoItemsVo.getDateParam()) {  
 startDate = DateUtils.addMonth(endDate, -6);  
 }  
 } else {  
 //默认当前日期前三个月记录  
 startDate = DateUtils.addMonth(endDate, -3);  
 //已完成  
 completedCNT = toDoItemsService.findItemsCount(toDoItemsVo.getUserName(), ToDoType.APPROVAL\_TYPE, "COMPLETED", startDate, endDate,  
 searchTodoTiers, searchExceptTodoTiers);  
 //未完成  
 notCompletedCNT = toDoItemsService.findItemsCount(toDoItemsVo.getUserName(), ToDoType.APPROVAL\_TYPE, "NOT\_COMPLETED", startDate, endDate,  
 searchTodoTiers, searchExceptTodoTiers);  
 //全部  
 allCNT = toDoItemsService.findItemsCount(toDoItemsVo.getUserName(), ToDoType.APPROVAL\_TYPE, "", startDate, endDate, searchTodoTiers,  
 searchExceptTodoTiers);  
 }  
 //返回结果  
 PageParam<IToDoItemsVo> pagerVo = new PageParam<IToDoItemsVo>();  
 //查询结果  
 toDoItemsVo.setToDoType(ToDoType.APPROVAL\_TYPE);  
 PageParam<Object> pager = toDoItemsService.findItemsByParamsNew(pageResult, toDoItemsVo, startDate, endDate, likeText, searchTodoTiers,  
 searchExceptTodoTiers);  
 pagerVo = this.toDoItemsPoToVoNew(pager);  
 dataMap.put("pagerVo", pagerVo);  
 dataMap.put("completedCNT", completedCNT);  
 dataMap.put("notCompletedCNT", notCompletedCNT);  
 dataMap.put("allCNT", allCNT);  
 return dataMap;  
 }  
 private PageParam<IToDoItemsVo> toDoItemsPoToVoNew(PageParam<Object> pageParams) {  
 PageParam<IToDoItemsVo> pageParam = new PageParam<IToDoItemsVo>();  
 List<IToDoItemsVo> toDoItemsVos = new ArrayList<IToDoItemsVo>();  
 if (pageParams != null && pageParams.getPageList() != null && pageParams.getPageList().size() > 0) {  
 //循环查询到的结果集  
 for (Object object : pageParams.getPageList()) {  
 Object[] obj = (Object[])object;  
 IToDoItemsVo iToDoItemsVo = new IToDoItemsVo();  
 String servicesType = (String) obj[0];  
 //基表字段  
 ItspServicesVo itspServicesVo = new ItspServicesVo();  
 if (null != obj[1]) {  
 itspServicesVo.setLastUpdateDttm((Date) obj[1]);  
 }  
 if (null != obj[2]) {  
 itspServicesVo.setSourceBusNo((String) obj[2]);  
 }  
 if (null != obj[3]) {  
 itspServicesVo.setDescription((String) obj[3]);  
 }  
 if (null != obj[4]) {  
 itspServicesVo.setCategory2((String) obj[4]);  
 }  
 if (null != obj[5]) {  
 itspServicesVo.setCategory3((String) obj[5]);  
 }  
 if (null != obj[6]) {  
 itspServicesVo.setDetailedDecription((String) obj[6]);  
 }  
 if (null != obj[7]) {  
 itspServicesVo.setCrtDttm((Date) obj[7]);  
 }  
 if (null != obj[8]) {  
 itspServicesVo.setSubmitterUserDisplayName((String) obj[8]);  
 }  
 if (null != obj[9]) {  
itspServicesVo.setPlatStatus(ItspServiceStatus.getItspServiceStatusByName2((String) obj[9]));  
 }  
 if (null != obj[10]) {  
 itspServicesVo.setBusNo((String) obj[10]);  
 }  
 if (null != obj[11]) {  
 itspServicesVo.setId((String) obj[11]);  
 }  
 //待办表字段  
 if (null != obj[12]) {  
 iToDoItemsVo.setToDoTier(ToDoTier.getToDoTierByName((String) obj[12]));  
 }  
 if (null != obj[13]) {  
 iToDoItemsVo.setToDoUrl((String) obj[13]);  
 }  
 if (null != obj[14]) {  
 iToDoItemsVo.setToDoId((String) obj[14]);  
 }  
 if (null != obj[15]) {  
 iToDoItemsVo.setId((String) obj[15]);  
 }  
 if (null != obj[16]) {  
 iToDoItemsVo.setToDoType(ToDoType.getToDoTypeByName((String) obj[16]));  
 }  
 if (null != obj[17]) {  
iToDoItemsVo.setToDoStatus(ToDoStatus.getToDoStatusByName((String)obj[17]));  
 }  
 //工作单  
 ItspRequestWorkOrderVo itspRequestWorkOrderVo = new ItspRequestWorkOrderVo();  
 if (null != obj[18]) {  
 itspRequestWorkOrderVo.setWorkNum((String)obj[18]);  
 }  
 if (null != obj[19]) {  
 itspRequestWorkOrderVo.setLastUpdateDttm((Date) obj[19]);  
 }  
 if (null != obj[20]) {  
 itspRequestWorkOrderVo.setSummary((String)obj[20]);  
 }  
 if (null != obj[21]) {  
 itspRequestWorkOrderVo.setDescription((String)obj[21]);  
 }  
 //按类型填入字段  
 if (servicesType.equals("INCIDENT") || servicesType.equals("REQUEST") || servicesType.equals("CHANGE") ||  
 servicesType.equals("RELEASE") || servicesType.equals("NEED") || servicesType.equals("FAULT") || servicesType.equals("CHECKINCIDENT")) {  
 if (servicesType.equals("INCIDENT")) {  
 if (null != obj[22]) {  
 itspServicesVo.setSystemName((String) obj[22]);  
 }  
 if (null != obj[23]) {  
 itspServicesVo.setPriority(PriorityType.getPriorityTypeByName((String) obj[23]));  
 }  
 if (null != obj[24]) {  
 itspServicesVo.setIsCheckIncident("1".equals(obj[24]+""));  
 }  
 }  
 if (servicesType.equals("REQUEST")) {  
 if (null != obj[25]) {  
 itspServicesVo.setRequestLevel((String) obj[25]);  
 }  
 if (null != obj[40]) {  
 itspServicesVo.setDescription((String) obj[40]);  
 }  
 if (null != (String)obj[41]) {  
 itspServicesVo.setDetailedDecription((String) obj[41]);  
 } else {  
 if (null != obj[42]) {  
 itspServicesVo.setDetailedDecription((String) obj[42]);  
 }  
 }   
 }  
 if (servicesType.equals("CHANGE")) {  
 if (null != obj[26]) {  
 itspServicesVo.setRiskLevel(RiskLevel.getRiskLevelByName((String) obj[26]));  
 }  
 }  
 if (servicesType.equals("RELEASE")) {  
 if (null != obj[27]) {  
 itspServicesVo.setRiskLevel(RiskLevel.getRiskLevelByName((String) obj[27]));  
 }  
 }  
 if (servicesType.equals("NEED")) {  
 if (null != obj[28]) {  
 itspServicesVo.setItspNeedId((String) obj[28]);  
 }  
 if (null != obj[29]) {  
 itspServicesVo.setItspNeedBusNo((String) obj[29]);  
 }  
 if (null != obj[30]) {  
 itspServicesVo.setNeedLevel(NeedLevel.selectByJiraValue((String) obj[30]));  
 }  
 if (null != obj[31]) {  
 itspServicesVo.setIsToSend("1".equals(obj[31]+""));  
 }  
 }  
 if (servicesType.equals("FAULT")) {  
 if (null != obj[32]) {  
 itspServicesVo.setFaultPriorityType(FaultPriorityType.getFaultPriorityTypeByName((String) obj[32]));  
 }  
 }  
 if (servicesType.equals("CHECKINCIDENT")) {  
 if (null != obj[33]) {  
 itspServicesVo.setBmcIncidentNum((String) obj[33]);  
 }  
 if (null != obj[34]) {  
 itspServicesVo.setSystemName((String) obj[34]);  
 }  
 if (null != obj[35]) {  
 itspServicesVo.setInfluenceNames((String) obj[35]);  
 }  
 if (null != obj[36]) {  
 itspServicesVo.setIncResponsibleNames((String) obj[36]);  
 }  
 if (null != obj[37]) {  
 itspServicesVo.setIncDescrption((String) obj[37]);  
 }  
 if (null != obj[38]) {  
 itspServicesVo.setIsCheckIncident("1".equals(obj[38]+""));  
 }  
 if (null != obj[39]) {  
itspServicesVo.setCheckIncType(CheckIncType.getCheckTypeByValue((String) obj[39]));  
 }  
 }  
 }  
 iToDoItemsVo.setiItspServicesVo(itspServicesVo);  
 iToDoItemsVo.setItspRequestWorkOrderVo(itspRequestWorkOrderVo);  
 toDoItemsVos.add(iToDoItemsVo);  
 }  
 pageParam.setPageList(toDoItemsVos);  
 pageParam.setPageNo(pageParams.getPageNo());  
 pageParam.setPageSize(pageParams.getPageSize());  
 pageParam.setTotalSize(pageParams.getTotalSize());  
 }  
 return pageParam;  
 }  
}

package com.hungraim.wx.httpinvoker.impl;  
import java.lang.reflect.InvocationTargetException;  
import java.util.ArrayList;  
import java.util.List;  
import javax.annotation.Resource;  
import org.apache.commons.beanutils.BeanUtils;  
import org.slf4j.Logger;  
import org.springframework.stereotype.Component;  
import com.hungraim.wx.base.page.PageParam;  
import com.hungraim.wx.base.util.LoggerUtil;  
import com.hungraim.wx.component.services.model.ItspBMCKeyTransaction;  
import com.hungraim.wx.component.services.model.ItspServiceRequestCatalogMapping;  
import com.hungraim.wx.component.services.model.ItspServicesCatalogMapping;  
import com.hungraim.wx.component.services.service.ItspBmckeyTransactionService;  
import com.hungraim.wx.component.services.service.ItspServiceRequestCatalogMappingService;  
import com.hungraim.wx.component.services.service.ItspServicesCatalogMappingService;  
import com.hungraim.wx.httpinvoker.inf.IBmcIncidentCatalogService;  
import com.hungraim.wx.httpinvoker.vo.ItspBmcKeyTransactionVo;  
import com.hungraim.wx.httpinvoker.vo.ItspServiceCatalogMappingVo;  
import com.hungraim.wx.httpinvoker.vo.ItspServiceRequestCatalogMappingVo;  
@Component  
public class BmcIncidentCatalogServiceImpl implements IBmcIncidentCatalogService {  
 Logger logger = LoggerUtil.getLogger();  
 @Resource  
 ItspServicesCatalogMappingService itspServicesCatalogMappingService;  
 @Resource  
 ItspServiceRequestCatalogMappingService itspServiceRequestCatalogMappingService;  
 @Resource  
 ItspBmckeyTransactionService itspBmckeyTransactionService;  
 @Override  
 public List<ItspServiceCatalogMappingVo> getDeptOneList(String category1Desc) {  
 List<ItspServiceCatalogMappingVo> listVo = new ArrayList<ItspServiceCatalogMappingVo>();  
 List<ItspServicesCatalogMapping> list = itspServicesCatalogMappingService.getDeptOneList(category1Desc);  
 if (null != list && list.size() > 0) {  
 for (ItspServicesCatalogMapping detail : list) {  
 ItspServiceCatalogMappingVo itspServiceCatalogMappingVo = new ItspServiceCatalogMappingVo();  
 try {  
 BeanUtils.copyProperties(itspServiceCatalogMappingVo, detail);  
 } catch (IllegalAccessException e) {  
 e.printStackTrace();  
 } catch (InvocationTargetException e) {  
 e.printStackTrace();  
 }  
 listVo.add(itspServiceCatalogMappingVo);  
 }  
 }  
 return listVo;  
 }  
 @Override  
 public List<ItspServiceCatalogMappingVo> getDeptTwoList(String category1Desc) {  
 List<ItspServiceCatalogMappingVo> listVo = new ArrayList<ItspServiceCatalogMappingVo>();  
 List<ItspServicesCatalogMapping> list = itspServicesCatalogMappingService.getDeptTwoList(category1Desc);  
 if (null != list && list.size() > 0) {  
 for (ItspServicesCatalogMapping detail : list) {  
 ItspServiceCatalogMappingVo itspServiceCatalogMappingVo = new ItspServiceCatalogMappingVo();  
 try {  
 BeanUtils.copyProperties(itspServiceCatalogMappingVo, detail);  
 } catch (IllegalAccessException e) {  
 e.printStackTrace();  
 } catch (InvocationTargetException e) {  
 e.printStackTrace();  
 }  
 listVo.add(itspServiceCatalogMappingVo);  
 }  
 }  
 return listVo;  
 }  
 @Override  
 public PageParam<ItspServiceCatalogMappingVo> getBmcIncidentCatalogList(PageParam<ItspServiceCatalogMappingVo> pageResult,  
 ItspServiceCatalogMappingVo itspServiceCatalogMappingVo) {  
 List<ItspServiceCatalogMappingVo> listVo = new ArrayList<ItspServiceCatalogMappingVo>();  
 PageParam<ItspServicesCatalogMapping> pageParam = itspServicesCatalogMappingService.getBmcIncidentCatalogList(pageResult,  
 itspServiceCatalogMappingVo);  
 PageParam<ItspServiceCatalogMappingVo> pagerVo = new PageParam<ItspServiceCatalogMappingVo>(pageParam);  
 if (null != pageParam && null != pageParam.getPageList() && pageParam.getPageList().size() > 0) {  
 for (ItspServicesCatalogMapping itspServicesCatalogMapping : pageParam.getPageList()) {  
 ItspServiceCatalogMappingVo itspServiceCatalogMappingVo1 = new ItspServiceCatalogMappingVo();  
 try {  
 BeanUtils.copyProperties(itspServiceCatalogMappingVo1, itspServicesCatalogMapping);  
 } catch (IllegalAccessException e) {  
 e.printStackTrace();  
 } catch (InvocationTargetException e) {  
 e.printStackTrace();  
 }  
 listVo.add(itspServiceCatalogMappingVo1);  
 }  
 pagerVo.setPageList(listVo);  
 }  
 return pagerVo;  
 }  
 @Override  
 public List<ItspServiceCatalogMappingVo> downLoadBmcIncidentList(String category1Desc, String category2Desc, String bmcIncNum) {  
 List<ItspServiceCatalogMappingVo> listVo = new ArrayList<ItspServiceCatalogMappingVo>();  
 List<ItspServicesCatalogMapping> list = itspServicesCatalogMappingService.downLoadBmcIncidentList(category1Desc, category2Desc, bmcIncNum);  
 if (null != list && list.size() > 0) {  
 for (ItspServicesCatalogMapping itspServicesCatalogMapping : list) {  
 ItspServiceCatalogMappingVo itspServiceCatalogMappingVo = new ItspServiceCatalogMappingVo();  
 try {  
 BeanUtils.copyProperties(itspServiceCatalogMappingVo, itspServicesCatalogMapping);  
 } catch (IllegalAccessException e) {  
 e.printStackTrace();  
 } catch (InvocationTargetException e) {  
 e.printStackTrace();  
 }  
 listVo.add(itspServiceCatalogMappingVo);  
 }  
 }  
 return listVo;  
 }  
 @Override  
 public PageParam<ItspBmcKeyTransactionVo> getBmcKeyTransactionList(PageParam<ItspBmcKeyTransactionVo> pageResult,  
 ItspBmcKeyTransactionVo itspBmcKeyTransactionVo) {  
 List<ItspBmcKeyTransactionVo> listVo = new ArrayList<ItspBmcKeyTransactionVo>();  
 PageParam<ItspBMCKeyTransaction> pageParam = itspBmckeyTransactionService.getBmcKeyTransactionList(pageResult, itspBmcKeyTransactionVo);  
 PageParam<ItspBmcKeyTransactionVo> pagerVo = new PageParam<ItspBmcKeyTransactionVo>(pageParam);  
 if (null != pageParam && null != pageParam.getPageList() && pageParam.getPageList().size() > 0) {  
 for (ItspBMCKeyTransaction itspBMCKeyTransaction : pageParam.getPageList()) {  
 ItspBmcKeyTransactionVo itspBmcKeyTransactionVo1 = new ItspBmcKeyTransactionVo();  
 try {  
 BeanUtils.copyProperties(itspBmcKeyTransactionVo1, itspBMCKeyTransaction);  
 } catch (IllegalAccessException e) {  
 e.printStackTrace();  
 } catch (InvocationTargetException e) {  
 e.printStackTrace();  
 }  
 listVo.add(itspBmcKeyTransactionVo1);  
 }  
 pagerVo.setPageList(listVo);  
 }  
 return pagerVo;  
 }  
 @Override  
 public List<ItspBmcKeyTransactionVo> getBmcKeyTransactionDeptOneList(String sourceCategory1) {  
 List<ItspBmcKeyTransactionVo> listVo = new ArrayList<ItspBmcKeyTransactionVo>();  
 List<ItspBMCKeyTransaction> list = itspBmckeyTransactionService.getBmcKeyTransactionDeptOneList(sourceCategory1);  
 if (null != list && list.size() > 0) {  
 for (ItspBMCKeyTransaction detail : list) {  
 ItspBmcKeyTransactionVo itspBmcKeyTransactionVo = new ItspBmcKeyTransactionVo();  
 try {  
 BeanUtils.copyProperties(itspBmcKeyTransactionVo, detail);  
 } catch (IllegalAccessException e) {  
 e.printStackTrace();  
 } catch (InvocationTargetException e) {  
 e.printStackTrace();  
 }  
 listVo.add(itspBmcKeyTransactionVo);  
 }  
 }  
 return listVo;  
 }  
 @Override  
 public List<ItspBmcKeyTransactionVo> getBmcKeyTransactionDeptTwoList(String sourceCategory1) {  
 List<ItspBmcKeyTransactionVo> listVo = new ArrayList<ItspBmcKeyTransactionVo>();  
 List<ItspBMCKeyTransaction> list = itspBmckeyTransactionService.getBmcKeyTransactionDeptTwoList(sourceCategory1);  
 if (null != list && list.size() > 0) {  
 for (ItspBMCKeyTransaction detail : list) {  
 ItspBmcKeyTransactionVo itspBmcKeyTransactionVo = new ItspBmcKeyTransactionVo();  
 try {  
 BeanUtils.copyProperties(itspBmcKeyTransactionVo, detail);  
 } catch (IllegalAccessException e) {  
 e.printStackTrace();  
 } catch (InvocationTargetException e) {  
 e.printStackTrace();  
 }  
 listVo.add(itspBmcKeyTransactionVo);  
 }  
 }  
 return listVo;  
 }  
 @Override  
 public List<ItspBmcKeyTransactionVo> downLoadBmcKeyTransactionList(String sourceCategory1, String sourceCategory2) {  
 List<ItspBmcKeyTransactionVo> listVo = new ArrayList<ItspBmcKeyTransactionVo>();  
 List<ItspBMCKeyTransaction> list = itspBmckeyTransactionService.downLoadBmcKeyTransactionList(sourceCategory1, sourceCategory2);  
 if (null != list && list.size() > 0) {  
 for (ItspBMCKeyTransaction itspBMCKeyTransaction : list) {  
 ItspBmcKeyTransactionVo itspBmcKeyTransactionVo = new ItspBmcKeyTransactionVo();  
 try {  
 BeanUtils.copyProperties(itspBmcKeyTransactionVo, itspBMCKeyTransaction);  
 } catch (IllegalAccessException e) {  
 e.printStackTrace();  
 } catch (InvocationTargetException e) {  
 e.printStackTrace();  
 }  
 listVo.add(itspBmcKeyTransactionVo);  
 }  
 }  
 return listVo;  
 }  
 @Override  
 public PageParam<ItspServiceRequestCatalogMappingVo> getBmcRequestQueryList(PageParam<ItspServiceRequestCatalogMappingVo> pageResult,  
 ItspServiceRequestCatalogMappingVo itspServiceRequestCatalogMappingVo) {  
 List<ItspServiceRequestCatalogMappingVo> listVo = new ArrayList<ItspServiceRequestCatalogMappingVo>();  
 PageParam<ItspServiceRequestCatalogMapping> pageParam = itspServiceRequestCatalogMappingService.getBmcRequestQueryList(pageResult,  
 itspServiceRequestCatalogMappingVo);  
 PageParam<ItspServiceRequestCatalogMappingVo> pagerVo = new PageParam<ItspServiceRequestCatalogMappingVo>(pageParam);  
 if (null != pageParam && null != pageParam.getPageList() && pageParam.getPageList().size() > 0) {  
 for (ItspServiceRequestCatalogMapping itspServiceRequestCatalogMapping : pageParam.getPageList()) {  
 ItspServiceRequestCatalogMappingVo itspServiceRequestCatalogMappingVo1 = new ItspServiceRequestCatalogMappingVo();  
 try {  
 BeanUtils.copyProperties(itspServiceRequestCatalogMappingVo1, itspServiceRequestCatalogMapping);  
 } catch (IllegalAccessException e) {  
 e.printStackTrace();  
 } catch (InvocationTargetException e) {  
 e.printStackTrace();  
 }  
 listVo.add(itspServiceRequestCatalogMappingVo1);  
 }  
 pagerVo.setPageList(listVo);  
 }  
 return pagerVo;  
 }  
 @Override  
 public List<ItspServiceRequestCatalogMappingVo> getClassifyOneByName(String category1) {  
 List<ItspServiceRequestCatalogMappingVo> listVo = new ArrayList<ItspServiceRequestCatalogMappingVo>();  
 List<ItspServiceRequestCatalogMapping> list = itspServiceRequestCatalogMappingService.getClassifyOneByName(category1);  
 if (null != list && list.size() > 0) {  
 for (ItspServiceRequestCatalogMapping detail : list) {  
 ItspServiceRequestCatalogMappingVo itspServiceRequestCatalogMappingVo = new ItspServiceRequestCatalogMappingVo();  
 try {  
 BeanUtils.copyProperties(itspServiceRequestCatalogMappingVo, detail);  
 } catch (IllegalAccessException e) {  
 e.printStackTrace();  
 } catch (InvocationTargetException e) {  
 e.printStackTrace();  
 }  
 listVo.add(itspServiceRequestCatalogMappingVo);  
 }  
 }  
 return listVo;  
 }  
 @Override  
 public List<ItspServiceRequestCatalogMappingVo> getClassifyTwoByName(String category1, String category2) {  
 List<ItspServiceRequestCatalogMappingVo> listVo = new ArrayList<ItspServiceRequestCatalogMappingVo>();  
 List<ItspServiceRequestCatalogMapping> list = itspServiceRequestCatalogMappingService.getClassifyTwoByName(category1, category2);  
 if (null != list && list.size() > 0) {  
 for (ItspServiceRequestCatalogMapping detail : list) {  
 ItspServiceRequestCatalogMappingVo itspServiceRequestCatalogMappingVo = new ItspServiceRequestCatalogMappingVo();  
 try {  
 BeanUtils.copyProperties(itspServiceRequestCatalogMappingVo, detail);  
 } catch (IllegalAccessException e) {  
 e.printStackTrace();  
 } catch (InvocationTargetException e) {  
 e.printStackTrace();  
 }  
 listVo.add(itspServiceRequestCatalogMappingVo);  
 }  
 }  
 return listVo;  
 }  
 @Override  
 public List<ItspServiceRequestCatalogMappingVo> downLoadBmcRequestList(String category1, String category2, String category3,  
 String serviceRequestName, String serviceRequestNo) {  
 List<ItspServiceRequestCatalogMappingVo> listVo = new ArrayList<ItspServiceRequestCatalogMappingVo>();  
 List<ItspServiceRequestCatalogMapping> list = itspServiceRequestCatalogMappingService.downLoadBmcRequestList(category1, category2, category3,  
 serviceRequestName, serviceRequestNo);  
 if (null != list && list.size() > 0) {  
 for (ItspServiceRequestCatalogMapping itspServiceRequestCatalogMapping : list) {  
 ItspServiceRequestCatalogMappingVo itspServiceRequestCatalogMappingVo = new ItspServiceRequestCatalogMappingVo();  
 try {  
 BeanUtils.copyProperties(itspServiceRequestCatalogMappingVo, itspServiceRequestCatalogMapping);  
 } catch (IllegalAccessException e) {  
 e.printStackTrace();  
 } catch (InvocationTargetException e) {  
 e.printStackTrace();  
 }  
 listVo.add(itspServiceRequestCatalogMappingVo);  
 }  
 }  
 return listVo;  
 }  
}