石化行业工艺文件交换系统

源代码

北京杰迅鸿翔信息技术有限公司

**IndexContorller.JAVA**

package com.jiexunnet.docexchange.web.controller;

import com.jiexunnet.common.web.filter.LoginIncerceptor;

import com.jiexunnet.usm.model.TUser;

import com.jiexunnet.usm.service.IUsmService;

import java.util.List;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpSession;

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

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

import org.springframework.stereotype.Controller;

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

import org.springframework.web.servlet.ModelAndView;

@Controller("indexController")

public class IndexController

{

@Autowired

@Qualifier("IUsmService")

private IUsmService usmService;

@RequestMapping({"/index"})

public String index(HttpServletRequest request)

{

return "login";

}

@RequestMapping({"/logout.do"})

public String loginOut(HttpSession session) {

return "login";

}

@RequestMapping({"/login.do"})

public ModelAndView login(String loginName, String password, HttpSession session) { ModelAndView mav = null;

TUser user = this.usmService.login(loginName, password);

if (user == null) {

mav = new ModelAndView("login");

mav.addObject("message", "用户名或密码错误");

return mav;

}

LoginIncerceptor loginIncerceptor = new LoginIncerceptor();

loginIncerceptor.putSessionMap(session, user);

List depts = this.usmService.getUserDepts(user);

session.setAttribute("user", user);

session.setAttribute("depts", depts);

mav = new ModelAndView("mainFrame");

return mav; }

@RequestMapping({"/goRightPage.do"})

public ModelAndView goRightPage() {

ModelAndView mav = new ModelAndView("/rightFrame");

return mav;

}

@RequestMapping({"/goBottomPage.do"})

public ModelAndView goBottomPage() {

ModelAndView mav = new ModelAndView("/bottomFrame");

return mav;

}

@RequestMapping({"/oa/index/oaIncludeJs.do"})

public String oaIncludeJs() {

return "oaIncludeJs";

}

@RequestMapping({"/topFrame"})

public String topFrame(HttpServletRequest request) { return "topFrame"; }

@RequestMapping({"/leftFrame"})

public String left(HttpServletRequest request) {

return "leftFrame";

}

@RequestMapping({"/rightFrame"})

public String right(HttpServletRequest request) {

return "rightFrame";

}

@RequestMapping({"/bottomFrame"})

public String bottom(HttpServletRequest request) { return "bottomFrame";

}

public IUsmService getUsmService()

{

return this.usmService;

}

public void setUsmService(IUsmService usmService)

{

this.usmService = usmService;

}

}

**SendController.java**

package com.jiexunnet.docexchange.web.controller;

import com.jiexunnet.docexchange.dao.DfsdlDAO;

import com.jiexunnet.docexchange.dao.IAmqpTempDao;

import com.jiexunnet.docexchange.dao.TblMsgFailureDao;

import com.jiexunnet.docexchange.dao.TblMsgNewSreelDao;

import com.jiexunnet.docexchange.model.TAmqpTemp;

import com.jiexunnet.docexchange.model.TBlackAndWhite;

import com.jiexunnet.docexchange.model.TBm;

import com.jiexunnet.docexchange.model.TBmPerson;

import com.jiexunnet.docexchange.model.TFileConventional;

import com.jiexunnet.docexchange.model.TIncept;

import com.jiexunnet.docexchange.model.TReply;

import com.jiexunnet.docexchange.model.TReplyFile;

import com.jiexunnet.docexchange.model.TSend;

import com.jiexunnet.docexchange.model.TSendFile;

import com.jiexunnet.docexchange.model.TZzbm;

import com.jiexunnet.docexchange.service.BlackAndWhiteService;

import com.jiexunnet.docexchange.service.GwjhAttachmentService;

import com.jiexunnet.docexchange.service.IBmPersonService;

import com.jiexunnet.docexchange.service.IBmService;

import com.jiexunnet.docexchange.service.IInceptService;

import com.jiexunnet.docexchange.service.IReplyService;

import com.jiexunnet.docexchange.service.ISendService;

import com.jiexunnet.docexchange.service.IZzbmService;

import com.jiexunnet.docexchange.service.ReplyAttachmentService;

import com.jiexunnet.common.pagination.IPageContext;

import com.jiexunnet.common.pagination.Page;

import com.jiexunnet.common.util.CatalogUtil;

import com.jiexunnet.usm.model.TDept;

import com.jiexunnet.usm.model.TUser;

import com.jiexunnet.usm.service.IDeptService;

import java.beans.PropertyEditorSupport;

import java.io.File;

import java.io.FileInputStream;

import java.io.FileNotFoundException;

import java.io.IOException;

import java.io.InputStream;

import java.io.OutputStream;

import java.io.PrintStream;

import java.io.PrintWriter;

import java.io.UnsupportedEncodingException;

import java.text.DateFormat;

import java.text.ParseException;

import java.text.SimpleDateFormat;

import java.util.ArrayList;

import java.util.Calendar;

import java.util.Collections;

import java.util.Comparator;

import java.util.Date;

import java.util.HashMap;

import java.util.Iterator;

import java.util.List;

import java.util.Map;

import java.util.Set;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

import javax.servlet.http.HttpSession;

import net.sf.json.JSONArray;

import org.apache.poi.hssf.usermodel.HSSFCell;

import org.apache.poi.hssf.usermodel.HSSFCellStyle;

import org.apache.poi.hssf.usermodel.HSSFRow;

import org.apache.poi.hssf.usermodel.HSSFSheet;

import org.apache.poi.hssf.usermodel.HSSFWorkbook;

import org.springframework.batch.core.Job;

import org.springframework.batch.core.JobParametersBuilder;

import org.springframework.batch.core.JobParametersInvalidException;

import org.springframework.batch.core.launch.JobLauncher;

import org.springframework.batch.core.repository.JobExecutionAlreadyRunningException;

import org.springframework.batch.core.repository.JobInstanceAlreadyCompleteException;

import org.springframework.batch.core.repository.JobRestartException;

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

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

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

import org.springframework.stereotype.Controller;

import org.springframework.web.bind.WebDataBinder;

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

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

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

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

import org.springframework.web.multipart.MultipartFile;

import org.springframework.web.multipart.MultipartHttpServletRequest;

import org.springframework.web.multipart.commons.CommonsMultipartFile;

import org.springframework.web.servlet.ModelAndView;

@Controller

public class SendController

{

@Autowired

@Qualifier("BlackAndWhiteService")

private BlackAndWhiteService blackandwhiteservice;

@Autowired

@Qualifier("IAmqpTempDao")

private IAmqpTempDao amqpTempDao;

@Autowired

@Qualifier("ISendService")

private ISendService sendService;

@Autowired

@Qualifier("IReplyService")

private IReplyService replyService;

@Autowired

@Qualifier("IInceptService")

private IInceptService inceptService;

@Autowired

@Qualifier("GwjhAttachmentService")

private GwjhAttachmentService attachmentService;

@Autowired

@Qualifier("ReplyAttachmentService")

private ReplyAttachmentService replyAttachmentService;

@Autowired

private TblMsgFailureDao tblmsgfaiuredao;

@Autowired

@Qualifier("IBmPersonService")

private IBmPersonService bmPersonService;

@Autowired

private TblMsgNewSreelDao tblmsgnewsreeldao;

@Autowired

private DfsdlDAO dfsdldao;

@Autowired

@Qualifier("IBmService")

private IBmService bmService;

@Autowired

@Qualifier("IDeptService")

private IDeptService deptService;

@Autowired

@Qualifier("IZzbmService")

private IZzbmService zzbmService;

@Autowired

JobLauncher jobLauncher;

@Autowired

Job job;

private String rootPath;

private String docexchangIp;

@RequestMapping({"/user/shouye.do"})

public ModelAndView shouye()

{

ModelAndView mav = new ModelAndView("/rightFrame");

return mav;

}

@RequestMapping({"/gwjh/createsend.do"})

public ModelAndView createsend() {

ModelAndView mav = new ModelAndView("/gwjh/createsend");

return mav;

}

@RequestMapping({"/gwjh/hassent.do"})

public ModelAndView hassent(Integer limit, Integer cpage, String title, Date startDate, Date endDate, HttpSession session) {

List depts = (List)session.getAttribute("depts");

Integer deptId = ((TDept)depts.get(0)).getId();

DateFormat format = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss");

Date starttime = null;

Date endtime = null;

if (limit == null) limit = Integer.valueOf(15);

if (cpage == null) {

cpage = new Integer(1);

}

int size = 0;

size = this.sendService.findTotalHasSend(deptId, title, startDate, endDate);

int gpage = (int)Math.ceil(size / limit.intValue());

if (cpage.intValue() <= 0) {

cpage = Integer.valueOf(1);

} else if (gpage == 0) {

cpage = Integer.valueOf(1);

gpage = 1;

} else if (cpage.intValue() > gpage) {

cpage = Integer.valueOf(gpage);

}

ModelAndView mav = new ModelAndView("/gwjh/hassent");

List tsendlist = this.sendService.findHasSendLists((cpage.intValue() - 1) \* limit.intValue(), limit, deptId, title, startDate, endDate);

List sends = new ArrayList();

Iterator i = tsendlist.iterator();

while (i.hasNext()) {

sends.add(TsendToMap((TSend)i.next()));

}

tsendlist = null;

mav.addObject("total", Integer.valueOf(size));

mav.addObject("rows", sends);

mav.addObject("cpage", cpage);

mav.addObject("gpage", Integer.valueOf(gpage));

mav.addObject("title", title);

mav.addObject("startDate", startDate);

mav.addObject("endDate", endDate);

return mav;

}

@RequestMapping({"/gwjh/hassent\_jsqk.do"})

public ModelAndView hassent\_jsqk(Integer limit, Integer cpage, Integer sendId, Integer type) { ModelAndView mav = new ModelAndView("/gwjh/hassent\_jsqk");

if (sendId == null) {

mav.addObject("success", Boolean.valueOf(false));

return mav;

}

if (limit == null) limit = Integer.valueOf(15);

if (cpage == null) {

cpage = new Integer(1);

}

int size = 0;

Integer total = this.sendService.findSendCountIncepts(sendId, type);

size = total.intValue();

int gpage = (int)Math.ceil(size / limit.intValue());

if (cpage.intValue() <= 0) {

cpage = Integer.valueOf(1);

} else if (gpage == 0) {

cpage = Integer.valueOf(1);

gpage = 1;

} else if (cpage.intValue() > gpage) {

cpage = Integer.valueOf(gpage);

}

List list = this.sendService.findSendIncepts(Integer.valueOf((cpage.intValue() - 1) \* limit.intValue()), limit, sendId.intValue(), type.intValue());

List incepts = new ArrayList();

Iterator i = list.iterator();

while (i.hasNext()) {

incepts.add(inceptToMap((TIncept)i.next()));

}

list = null;

mav.addObject("total", total);

mav.addObject("rows", incepts);

mav.addObject("cpage", cpage);

mav.addObject("gpage", Integer.valueOf(gpage));

mav.addObject("sendId", sendId);

mav.addObject("type", type);

return mav; }

@RequestMapping({"/gwjh/hassent\_ckhf.do"})

public ModelAndView hassent\_ckhf(Integer cpage, Integer limit, Integer sendId, Integer inceptId) {

ModelAndView mav = new ModelAndView("/gwjh/hassent\_ckhf");

if (limit == null) limit = Integer.valueOf(15);

if (cpage == null) {

cpage = new Integer(1);

}

int size = 0;

Page page1 = this.sendService.SelectSendReply((cpage.intValue() - 1) \* limit.intValue(), limit.intValue(), sendId, inceptId);

size = page1.getContext().getTotal();

int gpage = (int)Math.ceil(size / limit.intValue());

if (cpage.intValue() <= 0) {

cpage = Integer.valueOf(1);

} else if (gpage == 0) {

cpage = Integer.valueOf(1);

gpage = 1;

} else if (cpage.intValue() > gpage) {

cpage = Integer.valueOf(gpage);

}

List replys = new ArrayList();

Iterator i = page1.getItems().iterator();

while (i.hasNext()) {

replys.add(tReply2Map((TReply)i.next()));

}

mav.addObject("total", Integer.valueOf(size));

mav.addObject("rows", replys);

mav.addObject("cpage", cpage);

mav.addObject("gpage", Integer.valueOf(gpage));

mav.addObject("sendId", sendId);

mav.addObject("inceptId", inceptId);

return mav;

}

@RequestMapping({"/gwjh/hassent\_dxcs.do"})

public ModelAndView hassent\_dxcs(Integer limit, Integer cpage, String sendId, Integer type)

{

ModelAndView mav = new ModelAndView("/gwjh/hassent\_dxcs");

if (sendId == null) {

mav.addObject("success", Boolean.valueOf(false));

return mav;

}

if (limit == null) limit = Integer.valueOf(15);

if (cpage == null) {

cpage = new Integer(1);

}

int size = 0;

Integer total = this.sendService.findSendCountIncepts(Integer.valueOf(Integer.parseInt(sendId)), type);

size = total.intValue();

int gpage = (int)Math.ceil(size / limit.intValue());

if (cpage.intValue() <= 0) {

cpage = Integer.valueOf(1);

} else if (gpage == 0) {

cpage = Integer.valueOf(1);

gpage = 1;

} else if (cpage.intValue() > gpage) {

cpage = Integer.valueOf(gpage);

}

List list = this.sendService.findSendIncepts(Integer.valueOf((cpage.intValue() - 1) \* limit.intValue()), limit, Integer.parseInt(sendId), type.intValue());

List incepts = new ArrayList();

Iterator i = list.iterator();

while (i.hasNext()) {

incepts.add(inceptToMap((TIncept)i.next()));

}

list = null;

mav.addObject("success", Boolean.valueOf(true));

mav.addObject("rows", incepts);

mav.addObject("total", total);

mav.addObject("cpage", cpage);

mav.addObject("gpage", Integer.valueOf(gpage));

mav.addObject("sendId", sendId);

mav.addObject("type", type);

return mav;

}

@RequestMapping({"/gwjh/received.do"})

public ModelAndView received(Integer limit, Integer cpage, String title, Date startDate, Date endDate, HttpSession session) {

List depts = (List)session.getAttribute("depts");

Integer deptId = ((TDept)depts.get(0)).getId();

DateFormat format = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss");

if (limit == null) limit = Integer.valueOf(15);

if (cpage == null) {

cpage = new Integer(1);

}

int size = 0;

size = this.inceptService.queryReceivedCount(deptId, title, startDate, endDate);

int gpage = (int)Math.ceil(size / limit.intValue());

if (cpage.intValue() <= 0) {

cpage = Integer.valueOf(1);

} else if (gpage == 0) {

cpage = Integer.valueOf(1);

gpage = 1;

} else if (cpage.intValue() > gpage) {

cpage = Integer.valueOf(gpage);

}

ModelAndView mav = new ModelAndView("/gwjh/received");

List inceptlists = this.inceptService.queryReceived((cpage.intValue() - 1) \* limit.intValue(), limit, deptId, title, startDate, endDate);

List incepts = new ArrayList();

Iterator i = inceptlists.iterator();

while (i.hasNext()) {

incepts.add(inceptToMap((TIncept)i.next()));

}

inceptlists = null;

mav.addObject("total", Integer.valueOf(size));

mav.addObject("rows", incepts);

mav.addObject("cpage", cpage);

mav.addObject("gpage", Integer.valueOf(gpage));

mav.addObject("title", title);

mav.addObject("startDate", startDate);

mav.addObject("endDate", endDate);

return mav;

}

@RequestMapping({"/gwjh/unreceived.do"})

public ModelAndView unreceived(Integer limit, Integer cpage, String title, Date startDate, Date endDate, HttpSession session) {

List depts = (List)session.getAttribute("depts");

Integer deptId = ((TDept)depts.get(0)).getId();

DateFormat format = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss");

if (limit == null) limit = Integer.valueOf(15);

if (cpage == null) {

cpage = new Integer(1);

}

int size = 0;

size = this.inceptService.queryUnreceiveCount(deptId, title, startDate, endDate);

int gpage = (int)Math.ceil(size / limit.intValue());

if (cpage.intValue() <= 0) {

cpage = Integer.valueOf(1);

} else if (gpage == 0) {

cpage = Integer.valueOf(1);

gpage = 1;

} else if (cpage.intValue() > gpage) {

cpage = Integer.valueOf(gpage);

}

ModelAndView mav = new ModelAndView("/gwjh/unreceived");

List inceptlists = this.inceptService.queryUnreceiveList((cpage.intValue() - 1) \* limit.intValue(), limit, deptId, title, startDate, endDate);

List incepts = new ArrayList();

Iterator i = inceptlists.iterator();

while (i.hasNext()) {

incepts.add(inceptToMap((TIncept)i.next()));

}

inceptlists = null;

mav.addObject("total", Integer.valueOf(size));

mav.addObject("rows", incepts);

mav.addObject("cpage", cpage);

mav.addObject("gpage", Integer.valueOf(gpage));

mav.addObject("title", title);

mav.addObject("startDate", startDate);

mav.addObject("endDate", endDate);

return mav;

}

@RequestMapping({"/gwjh/qs.do"})

public ModelAndView qs() {

ModelAndView mav = new ModelAndView("/gwjh/received");

return mav;

}

@RequestMapping({"/gwjh/hassent\_bmry.do"})

public ModelAndView hassent\_bmry(Integer limit, Integer cpage, String sendId, Integer bmFlag)

{

if (limit == null) limit = Integer.valueOf(15);

if (cpage == null) {

cpage = new Integer(1);

}

int size = 0;

int gpage = (int)Math.ceil(size / limit.intValue());

if (cpage.intValue() <= 0) {

cpage = Integer.valueOf(1);

} else if (gpage == 0) {

cpage = Integer.valueOf(1);

gpage = 1;

} else if (cpage.intValue() > gpage) {

cpage = Integer.valueOf(gpage);

}

ModelAndView mav = new ModelAndView("/gwjh/hassent\_bmry");

List bmpersons = new ArrayList();

if (sendId != null) {

Integer sendId0 = Integer.valueOf(Integer.parseInt(sendId));

List bmbean = this.bmService.getBySendId(sendId0.intValue());

if (bmbean != null) {

for (int i = 0; i < bmbean.size(); i++)

bmpersons.add(bmPerson2Map((TBm)bmbean.get(i)));

}

}

mav.addObject("rows", bmpersons);

mav.addObject("totalBmPerson", Integer.valueOf(bmpersons.size()));

mav.addObject("sendId", sendId);

mav.addObject("cpage", cpage);

mav.addObject("gpage", Integer.valueOf(gpage));

if (bmFlag == null)

mav.addObject("bmFlag", Integer.valueOf(2));

else if (bmFlag.intValue() == 0)

mav.addObject("bmFlag", Integer.valueOf(0));

else if (bmFlag.intValue() == 1) {

mav.addObject("bmFlag", Integer.valueOf(1));

}

return mav;

}

@RequestMapping({"/gwjh/ckbm.do"})

public ModelAndView ckbm(Integer limit, Integer cpage, String sendId, HttpSession session) {

List depts = (List)session.getAttribute("depts");

Integer deptId = ((TDept)depts.get(0)).getId();

if (limit == null) limit = Integer.valueOf(15);

if (cpage == null) {

cpage = new Integer(1);

}

int size = 0;

int gpage = (int)Math.ceil(size / limit.intValue());

if (cpage.intValue() <= 0) {

cpage = Integer.valueOf(1);

} else if (gpage == 0) {

cpage = Integer.valueOf(1);

gpage = 1;

} else if (cpage.intValue() > gpage) {

cpage = Integer.valueOf(gpage);

}

ModelAndView mav = new ModelAndView("/gwjh/ckbm");

List bmpersons = new ArrayList();

if (sendId != null) {

List bmbean = this.bmService.getTBm(Integer.parseInt(sendId), deptId.intValue());

if (bmbean != null) {

for (int i = 0; i < bmbean.size(); i++)

bmpersons.add(bmPerson2Map((TBm)bmbean.get(i)));

}

}

mav.addObject("rows", bmpersons);

mav.addObject("sendId", sendId);

mav.addObject("cpage", cpage);

mav.addObject("gpage", Integer.valueOf(gpage));

return mav;

}

@RequestMapping({"/docexchange/send/send.do"})

public ModelAndView createSend(String inceptDeptIds, String flag, String alertformcontent, String atcuuid, @ModelAttribute("send") TSend send, HttpServletResponse response, HttpSession session)

throws IllegalStateException, IOException

{

ModelAndView mav = new ModelAndView("redirect:/gwjh/hassent.do");

ModelAndView login = new ModelAndView("/login");

ModelAndView error = new ModelAndView("/error\_all");

TUser user = (TUser)session.getAttribute("user");

if (user == null) {

return login;

}

send.setSendor(user.getUserName());

send.setSendorId(user.getId().intValue());

TDept sendDept = (TDept)this.deptService.get(Integer.valueOf(send.getSendDeptId()));

send.setSendServerId(sendDept.getActivemqQueueId());

send.setSendTime(new Date());

List flist = this.attachmentService.findAtcList(atcuuid);

String[] inceptDeptStrs = inceptDeptIds.split(",");

List inceptIdList = new ArrayList();

if ((inceptDeptStrs != null) && (inceptDeptStrs.length > 0)) {

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

inceptIdList.add(Integer.valueOf(Integer.parseInt(inceptDeptStrs[i])));

}

}

send.setTotleSend(Integer.valueOf(inceptIdList.size()));

send.setReceived(Integer.valueOf(0));

send.setUuid(atcuuid);

this.sendService.addSend(send, flist, inceptIdList, flag, alertformcontent);

return mav;

}

@RequestMapping({"docexchange/send/resend.do"})

@ResponseBody

public ModelAndView resend(Integer sendId, String inceptDeptIds, String flag, String alertformcontent) {

if ((inceptDeptIds != null) && (inceptDeptIds != "")) {

String[] ids = inceptDeptIds.split(",");

List idlist = new ArrayList();

if ((ids != null) && (ids.length > 0)) {

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

Integer id = Integer.valueOf(Integer.parseInt(ids[i]));

idlist.add(id);

}

this.sendService.resend(sendId, idlist, flag, alertformcontent);

}

}

ModelAndView mav = new ModelAndView("redirect:/gwjh/hassent.do");

return mav;

}

@RequestMapping({"/gwjh/unreceived\_ckxx.do"})

public ModelAndView unreceivedckxx(Integer sendId, Integer inceptId, Integer unreceiveTotal, HttpSession session) {

TUser user = (TUser)session.getAttribute("user");

TSend send = (TSend)this.sendService.get(sendId);

if (send != null) {

TDept tDept = (TDept)this.deptService.get(Integer.valueOf(send.getSendDeptId()));

if (tDept != null) {

TIncept tIncept = (TIncept)this.inceptService.get(inceptId);

if ((tIncept != null) &&

(tIncept.getCancelFlag() == 0)) {

this.sendService.doSign(sendId, inceptId, user);

TDept inceptDept = (TDept)this.deptService.get(Integer.valueOf(tIncept.getInceptDeptId()));

if ((inceptDept != null) &&

(tDept.getActivemqQueueId() != null) && (inceptDept.getActivemqQueueId() != null) &&

(inceptDept.getActivemqQueueId().equals(tDept.getActivemqQueueId())) &&

(tDept.getAmqpFlag() != null) && (tDept.getAmqpFlag().intValue() == 2)) {

startJob(send.getUuid(), inceptId, null, "sign");

}

}

}

}

ModelAndView mav = new ModelAndView("/gwjh/unreceived\_ckxx");

if (send.getId() != null) {

mav.addObject("sendId", send.getId());

mav.addObject("id", send.getId());

}

if (inceptId != null) {

mav.addObject("inceptId", inceptId);

}

if (send.getSendor() != null) {

mav.addObject("sendor", send.getSendor());

}

mav.addObject("sendorid", Integer.valueOf(send.getSendorId()));

if (send.getSendDept() != null) {

mav.addObject("sendDept", send.getSendDept());

}

if (send.getSendDept() != null) {

mav.addObject("sendDeptId", send.getSendDept());

}

if (send.getSendTitle() != null) {

mav.addObject("sendTitle", send.getSendTitle());

}

if (send.getSendType() != null) {

mav.addObject("sendType", send.getSendType());

}

mav.addObject("sendTime", new SimpleDateFormat("yyyy-MM-dd HH:mm:ss")

.format(send

.getSendTime()));

mav.addObject("sendTime1", new SimpleDateFormat("yyyy-MM-dd HH:mm:ss")

.format(send

.getSendTime()));

if (send.getSendKeyWord() != null) {

mav.addObject("sendKeyWord", send.getSendKeyWord());

}

if (send.getSendTitle1() != null) {

mav.addObject("sendTitle1", send.getSendTitle1());

}

if (send.getSendMj() != null) {

mav.addObject("sendMj", send.getSendMj());

}

if (send.getSendJjcd() != null) {

mav.addObject("sendJjcd", send.getSendJjcd());

}

if (send.getSendCopy() != null) {

mav.addObject("sendCopy", send.getSendCopy());

}

if (send.getSendSignDept() != null) {

mav.addObject("sendSignDept", send.getSendSignDept());

}

if (send.getSendSigner() != null) {

mav.addObject("sendSigner", send.getSendSigner());

}

if (send.getSendSignTime() != null)

mav.addObject("sendSignTime", new SimpleDateFormat("yyyy-MM-dd")

.format(send

.getSendSignTime()));

else

mav.addObject("sendSignTime", send.getSendSignTime());

if (send.getSendLxr() != null) {

mav.addObject("sendLxr", send.getSendLxr());

}

if (send.getSendLxdh() != null) {

mav.addObject("sendLxdh", send.getSendLxdh());

}

if (send.getSendPages() != null) {

mav.addObject("sendPages", send.getSendPages());

}

Set incepts = send.getIncepts();

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

Iterator item = incepts.iterator();

String inceptIds = "";

String inceptDepts = "";

while (item.hasNext()) {

TIncept incept = (TIncept)item.next();

Integer deptId = Integer.valueOf(incept.getInceptDeptId());

String inceptDept = incept.getInceptDept();

inceptDepts = inceptDepts + inceptDept;

inceptDepts = inceptDepts + ",";

inceptIds = inceptIds + deptId.toString();

inceptIds = inceptIds + ",";

}

if (inceptIds.length() > 1) {

inceptIds = inceptIds.substring(0, inceptIds.length() - 1);

}

if (inceptDepts.length() > 1) {

inceptDepts = inceptDepts.substring(0, inceptDepts.length() - 1);

}

System.out.println(inceptDepts);

mav.addObject("inceptIds", inceptIds);

mav.addObject("inceptDepts", inceptDepts);

}

TZzbm zzbm = this.zzbmService.findBySendId(sendId);

if (zzbm != null) {

mav.addObject("flag", zzbm.getFlag());

}

mav.addObject("sendScope", send.getSendScope());

mav.addObject("sendWh", send.getSendWh());

mav.addObject("unreceiveTotal", unreceiveTotal);

return mav;

}

@RequestMapping({"/gwjh/bm.do"})

public ModelAndView bm(String sendId, String inceptId, String flag) {

ModelAndView mav = new ModelAndView("/gwjh/bm");

ModelAndView zzbm = new ModelAndView("/gwjh/zzbm");

mav.addObject("sendId", sendId);

mav.addObject("inceptId", inceptId);

mav.addObject("flag", flag);

zzbm.addObject("sendId", sendId);

zzbm.addObject("inceptId", inceptId);

zzbm.addObject("flag", flag);

if (flag.equals("1")) {

return zzbm;

}

return mav;

}

@RequestMapping({"/docexchange/bmform/save.do"})

public String saveBMForm(String bmPersons, String ids, @ModelAttribute("Bm") TBm bm)

throws IllegalStateException, IOException

{

if (bm != null) {

this.bmService.saveBms(bm, ids, bmPersons);

if (bm.getSendId() != null) {

TSend tSend = (TSend)this.sendService.get(bm.getSendId());

if (tSend != null) {

TDept tDept = (TDept)this.deptService.get(Integer.valueOf(tSend.getSendDeptId()));

if ((tDept != null) &&

(bm.getDeptId() != null)) {

TDept bmDept = (TDept)this.deptService.get(bm.getDeptId());

if ((bmDept != null) &&

(bmDept.getActivemqQueueId() != null) && (tDept.getActivemqQueueId() != null) &&

(bmDept.getActivemqQueueId().equals(tDept.getActivemqQueueId())) &&

(tDept.getAmqpFlag() != null) && (tDept.getAmqpFlag().intValue() == 2)) {

startJob(tSend.getUuid(), bm.getInceptid(), bm.getUuid(), "bm");

}

}

}

}

}

return "redirect:/gwjh/received.do";

}

@RequestMapping({"/gwjh/hf.do"})

public ModelAndView hf(String inceptId) {

ModelAndView mav = new ModelAndView("/gwjh/hf");

mav.addObject("inceptId", inceptId);

return mav;

}

@RequestMapping({"/gwjh/unreceive\_hf.do"})

public ModelAndView unreceive\_hf(String inceptid) { ModelAndView mav = new ModelAndView("/gwjh/unreceive\_hf");

mav.addObject("inceptId", inceptid);

return mav; }

@RequestMapping({"/gwjh/jq.do"})

public ModelAndView jq(String inceptid) {

ModelAndView mav = new ModelAndView("/gwjh/jq");

mav.addObject("inceptid", inceptid);

return mav;

}

@RequestMapping({"/gwjh/hassent\_ckxx.do"})

public ModelAndView hassent\_ckxx(String sendId) {

ModelAndView mav = new ModelAndView("/gwjh/hassent\_ckxx");

TSend send = this.sendService.findSend(Integer.valueOf(sendId).intValue());

mav.addObject("sendId", send.getId());

mav.addObject("id", send.getId());

mav.addObject("uuid", send.getUuid());

mav.addObject("sendor", send.getSendor());

mav.addObject("sendorid", Integer.valueOf(send.getSendorId()));

mav.addObject("sendDept", send.getSendDept());

mav.addObject("sendDeptId", send.getSendDept());

mav.addObject("sendTitle", send.getSendTitle());

mav.addObject("sendType", send.getSendType());

mav.addObject("sendTime", new SimpleDateFormat("yyyy-MM-dd HH:mm:ss")

.format(send

.getSendTime()));

mav.addObject("sendTime1", new SimpleDateFormat("yyyy-MM-dd HH:mm:ss")

.format(send

.getSendTime()));

mav.addObject("sendKeyWord", send.getSendKeyWord());

mav.addObject("sendTitle1", send.getSendTitle1());

mav.addObject("sendMj", send.getSendMj());

mav.addObject("sendJjcd", send.getSendJjcd());

mav.addObject("sendCopy", send.getSendCopy());

mav.addObject("sendSignDept", send.getSendSignDept());

mav.addObject("sendSigner", send.getSendSigner());

if (send.getSendSignTime() != null)

mav.addObject("sendSignTime", new SimpleDateFormat("yyyy-MM-dd HH:mm:ss")

.format(send

.getSendSignTime()));

else

mav.addObject("sendSignTime", send.getSendSignTime());

mav.addObject("sendLxr", send.getSendLxr());

mav.addObject("sendLxdh", send.getSendLxdh());

mav.addObject("sendPages", send.getSendPages());

Set incepts = send.getIncepts();

Iterator item = incepts.iterator();

String inceptIds = "";

String inceptDepts = "";

while (item.hasNext()) {

TIncept incept = (TIncept)item.next();

Integer deptId = Integer.valueOf(incept.getInceptDeptId());

String inceptDept = incept.getInceptDept();

inceptDepts = inceptDepts + inceptDept;

inceptDepts = inceptDepts + ",";

inceptIds = inceptIds + deptId.toString();

inceptIds = inceptIds + ",";

}

if (inceptIds.length() > 1) {

inceptIds = inceptIds.substring(0, inceptIds.length() - 1);

}

if (inceptDepts.length() > 1) {

inceptDepts = inceptDepts.substring(0, inceptDepts.length() - 1);

}

mav.addObject("inceptDeptIds", inceptIds);

mav.addObject("inceptDepts", inceptDepts);

mav.addObject("sendScope", send.getSendScope());

mav.addObject("sendWh", send.getSendWh());

return mav;

}

@RequestMapping({"/gwjh/received\_ckxx.do"})

public ModelAndView receivedckxx(Integer sendId, Integer inceptId, String flag, HttpSession session) {

TUser user = (TUser)session.getAttribute("user");

this.sendService.doSign(sendId, inceptId, user);

ModelAndView mav = new ModelAndView("/gwjh/received\_ckxx");

TSend send = this.sendService.findSend(Integer.valueOf(sendId.intValue()).intValue());

mav.addObject("sendId", send.getId());

mav.addObject("inceptId", inceptId);

mav.addObject("flag", flag);

mav.addObject("sendor", send.getSendor());

mav.addObject("sendorid", Integer.valueOf(send.getSendorId()));

mav.addObject("sendDept", send.getSendDept());

mav.addObject("sendDeptId", send.getSendDept());

mav.addObject("sendTitle", send.getSendTitle());

mav.addObject("sendType", send.getSendType());

mav.addObject("sendTime", new SimpleDateFormat("yyyy-MM-dd HH:mm:ss")

.format(send

.getSendTime()));

mav.addObject("sendTime1", new SimpleDateFormat("yyyy-MM-dd HH:mm:ss")

.format(send

.getSendTime()));

mav.addObject("sendKeyWord", send.getSendKeyWord());

mav.addObject("sendTitle1", send.getSendTitle1());

mav.addObject("sendMj", send.getSendMj());

mav.addObject("sendJjcd", send.getSendJjcd());

mav.addObject("sendCopy", send.getSendCopy());

System.out.println(new SimpleDateFormat("yyyy-MM-dd HH:mm:ss")

.format(send

.getSendTime()) + "sendtime1");

mav.addObject("sendSignDept", send.getSendSignDept());

mav.addObject("sendSigner", send.getSendSigner());

if (send.getSendSignTime() != null)

mav.addObject("sendSignTime", new SimpleDateFormat("yyyy-MM-dd")

.format(send

.getSendSignTime()));

else

mav.addObject("sendSignTime", send.getSendSignTime());

mav.addObject("sendLxr", send.getSendLxr());

mav.addObject("sendLxdh", send.getSendLxdh());

mav.addObject("sendPages", send.getSendPages());

Set incepts = send.getIncepts();

Iterator item = incepts.iterator();

String inceptIds = "";

String inceptDepts = "";

while (item.hasNext()) {

TIncept incept = (TIncept)item.next();

Integer deptId = Integer.valueOf(incept.getInceptDeptId());

String inceptDept = incept.getInceptDept();

inceptDepts = inceptDepts + inceptDept;

inceptDepts = inceptDepts + ",";

inceptIds = inceptIds + deptId.toString();

inceptIds = inceptIds + ",";

}

if (inceptIds.length() > 1) {

inceptIds = inceptIds.substring(0, inceptIds.length() - 1);

}

if (inceptDepts.length() > 1) {

inceptDepts = inceptDepts.substring(0, inceptDepts.length() - 1);

}

mav.addObject("inceptIds", inceptIds);

mav.addObject("inceptDepts", inceptDepts);

mav.addObject("sendScope", send.getSendScope());

mav.addObject("sendWh", send.getSendWh());

return mav;

}

@RequestMapping({"/gwjh/depttreewithcheckbox.do"})

@ResponseBody

public Map<String, Object> gwjhDeptTree(String deptName) { List list = new ArrayList();

List deptLists = this.deptService.findChildrens(deptName);

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

for (int i = 0; i < deptLists.size(); i++) {

if ((((TDept)deptLists.get(i)).getId() != null) && (((TDept)deptLists.get(i)).getId().intValue() == 0))

{

continue;

}

Map root = new HashMap();

Integer parentId = null;

if (((TDept)deptLists.get(i)).getParentDept() != null) {

parentId = ((TDept)deptLists.get(i)).getParentDept().getId();

}

root.put("pId", parentId);

root.put("id", ((TDept)deptLists.get(i)).getId());

root.put("name", ((TDept)deptLists.get(i)).getDeptName());

list.add(root);

}

}

JSONArray jsonObject = JSONArray.fromObject(list);

Map mapTemp = new HashMap();

mapTemp.put("orgPersonTree", jsonObject.toString());

return mapTemp;

}

@RequestMapping({"/gwjh/cydepttreewithcheckbox.do"})

@ResponseBody

public Map<String, Object> cyDeptTree(HttpSession session)

{

List depts = (List)session.getAttribute("depts");

Integer deptId = ((TDept)depts.get(0)).getId();

List list = new ArrayList();

List roots = this.deptService.getcyRootDept(deptId);

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

for (int i = 0; i < roots.size(); i++)

{

Map root = new HashMap();

root.put("id", ((TFileConventional)roots.get(i)).getId());

root.put("name", ((TFileConventional)roots.get(i)).getDeptName());

List childCyDeptList = ((TFileConventional)roots.get(i)).getDepts();

for (TFileConventional childConventional : childCyDeptList) {

Map child = new HashMap();

child.put("pId", ((TFileConventional)roots.get(i)).getId());

child.put("id", childConventional.getId());

child.put("name", childConventional.getDeptName());

list.add(child);

}

list.add(root);

}

}

JSONArray jsonObject = JSONArray.fromObject(list);

Map mapTemp = new HashMap();

mapTemp.put("orgPersonTree", jsonObject.toString());

return mapTemp;

}

@ResponseBody

public List<Map<String, Object>> cytreewithcheckbox(HttpSession session) {

List depts = (List)session.getAttribute("depts");

Integer deptId = ((TDept)depts.get(0)).getId();

Map map = new HashMap();

List lists = new ArrayList();

try

{

List roots = this.deptService.getcyRootDept(deptId);

List list = new ArrayList(0);

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

for (int i = 0; i < roots.size(); i++) {

list.add(cyprocessChildwithCheckbox((TFileConventional)roots.get(i), deptId));

}

}

map.put("open", Boolean.valueOf(true));

map.put("name", "常用单位");

map.put("id", Integer.valueOf(-1));

if (list.size() > 1) {

map.put("hasChildren", Boolean.valueOf(true));

}

map.put("expanded", Boolean.valueOf(true));

map.put("children", list);

map.put("success", Boolean.valueOf(true));

lists.add(map);

} catch (Exception e) {

e.printStackTrace();

map.put("success", Boolean.valueOf(false));

}

return lists;

}

private Map<String, Object> cyprocessChildwithCheckbox(TFileConventional dept, Integer deptId) {

Map map = new HashMap();

List list = new ArrayList(0);

map.put("id", dept.getDeptId());

map.put("flag", dept.getDeptFlag());

map.put("name", dept.getDeptName());

map.put("expanded", Boolean.valueOf(true));

List children = dept.getDepts();

List TBlackAndWhitelist = this.blackandwhiteservice.getBlackAndWhite();

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

map.put("hasChildren", Boolean.valueOf(true));

for (int i = 0; i < children.size(); i++) {

boolean flag = true;

for (int k = 0; k < TBlackAndWhitelist.size(); k++)

{

int deptId1 = ((TFileConventional)children.get(i)).getId().intValue();

int deptId2 = ((TBlackAndWhite)TBlackAndWhitelist.get(k)).getDeptId().intValue();

int deptId3 = ((TBlackAndWhite)TBlackAndWhitelist.get(k)).getParentId().intValue();

if ((deptId.intValue() != deptId2) ||

(deptId1 != deptId3)) continue;

flag = false;

}

if (flag) {

list.add(cyprocessChildwithCheckbox((TFileConventional)children.get(i), deptId));

}

}

map.put("leaf", Boolean.valueOf(false));

map.put("open", Boolean.valueOf(true));

} else {

map.put("hasChildren", Boolean.valueOf(false));

}

if (list.size() > 0) {

map.put("children", list);

}

return map;

}

private Map<String, Object> processChildwithCheckbox(TDept dept, Integer deptId) {

Map map = new HashMap();

List list = new ArrayList(0);

map.put("id", dept.getId());

map.put("name", dept.getDeptName());

map.put("expanded", Boolean.valueOf(true));

List children = dept.getDepts();

List TBlackAndWhitelist = this.blackandwhiteservice.getBlackAndWhite();

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

map.put("hasChildren", Boolean.valueOf(true));

for (int i = 0; i < children.size(); i++) {

boolean flag = true;

for (int k = 0; k < TBlackAndWhitelist.size(); k++)

{

int deptId1 = ((TDept)children.get(i)).getId().intValue();

int deptId2 = ((TBlackAndWhite)TBlackAndWhitelist.get(k)).getDeptId().intValue();

int deptId3 = ((TBlackAndWhite)TBlackAndWhitelist.get(k)).getParentId().intValue();

if ((deptId.intValue() != deptId2) ||

(deptId1 != deptId3)) continue;

flag = false;

}

if ((((TDept)children.get(i)).getActivemqQueueId() != null) && (!((TDept)children.get(i)).getActivemqQueueId().equals("chaozhoushi"))) {

continue;

}

if (flag) {

list.add(processChildwithCheckbox((TDept)children.get(i), deptId));

}

}

map.put("leaf", Boolean.valueOf(false));

map.put("open", Boolean.valueOf(true));

} else {

map.put("hasChildren", Boolean.valueOf(false));

}

if (list.size() > 0) {

map.put("children", list);

}

return map;

}

@RequestMapping({"/docexchange/send/sendMonitorList.do"})

@ResponseBody

public Map<String, Object> sendMonitorlist(int page, int start, int limit, String title, Date startDate, Date endDate, HttpSession session) { Map map = new HashMap();

List depts = (List)session.getAttribute("depts");

Integer deptId = ((TDept)depts.get(0)).getId();

Page page1 = this.sendService.findSendMonitorlist(page, limit, deptId

.intValue(), title, startDate, endDate);

List tsendlist = page1.getItems();

List sends = new ArrayList();

Iterator i = tsendlist.iterator();

while (i.hasNext()) {

sends.add(TsendToMap((TSend)i.next()));

}

tsendlist = null;

map.put("success", Boolean.valueOf(true));

map.put("total", Integer.valueOf(page1.getContext().getTotal()));

map.put("data", sends);

return map; }

@RequestMapping({"/docexchange/send/backlist.do"})

@ResponseBody

public Map<String, Object> sendBacklist(int page, int start, int limit, String title, Date startDate, Date endDate, HttpSession session)

{

Map map = new HashMap();

List depts = (List)session.getAttribute("depts");

Integer deptId = ((TDept)depts.get(0)).getId();

Page page1 = this.sendService.findBacklist(page, limit, deptId.intValue(), title, startDate, endDate);

List tsendlist = page1.getItems();

List sends = new ArrayList();

Iterator i = tsendlist.iterator();

while (i.hasNext()) {

sends.add(TsendToMap((TSend)i.next()));

}

tsendlist = null;

map.put("success", Boolean.valueOf(true));

map.put("total", Integer.valueOf(page1.getContext().getTotal()));

map.put("data", sends);

return map;

}

@RequestMapping({"/docexchange/send/cancellist.do"})

@ResponseBody

public Map<String, Object> sendCancellist(int page, int start, int limit, String title, Date startDate, Date endDate, HttpSession session) {

Map map = new HashMap();

List depts = (List)session.getAttribute("depts");

Integer deptId = ((TDept)depts.get(0)).getId();

Page page1 = this.sendService.findCancellist(page, limit, deptId

.intValue(), title, startDate, endDate);

List tsendlist = page1.getItems();

List sends = new ArrayList();

Iterator i = tsendlist.iterator();

while (i.hasNext()) {

sends.add(TsendToMap((TSend)i.next()));

}

tsendlist = null;

map.put("success", Boolean.valueOf(true));

map.put("total", Integer.valueOf(page1.getContext().getTotal()));

map.put("data", sends);

return map;

}

@RequestMapping({"/docexchange/send/replylistfiles.do"})

@ResponseBody

public Map<String, Object> replylistfiles(Integer replyid) {

Map map = new HashMap();

if (replyid == null) {

map.put("success", Boolean.valueOf(false));

return map;

}

List files = this.replyService.findReplyfiles(replyid.intValue());

List list = new ArrayList();

Iterator i = files.iterator();

while (i.hasNext()) {

list.add(replyFileToMap((TReplyFile)i.next()));

}

files = null;

map.put("success", Boolean.valueOf(true));

map.put("data", list);

return map;

}

private Map<String, Object> replyFileToMap(TReplyFile file) {

Map map = new HashMap();

map.put("id", file.getId());

map.put("fileName", file.getFileName());

map.put("fileType", file.getFileType());

map.put("fileSize", Integer.valueOf(file.getFileSize()));

map.put("fileAddress", file.getFileAddress());

return map;

}

@RequestMapping({"docexchange/send/listreply.do"})

@ResponseBody

public Map<String, Object> listreply(Integer start, Integer limit, Integer sendId, Integer inceptId) { Map map = new HashMap();

Page page1 = this.sendService.SelectSendReply(start.intValue(), limit.intValue(), sendId, inceptId);

List replys = new ArrayList();

Iterator i = page1.getItems().iterator();

while (i.hasNext()) {

replys.add(tReply2Map((TReply)i.next()));

}

map.put("total", Integer.valueOf(page1.getContext().getTotal()));

map.put("rows", replys);

return map; }

@RequestMapping({"docexchange/send/listnotreply.do"})

@ResponseBody

public Map<String, Object> listnotreply(int page, int start, int limit, Integer sendId, Integer inceptId) {

Map map = new HashMap();

Page page1 = this.sendService.SelectSendnotReply(page, limit, sendId);

List replys = new ArrayList();

Iterator i = page1.getItems().iterator();

while (i.hasNext()) {

replys.add(tIncept2Map((TIncept)i.next()));

}

map.put("success", Boolean.valueOf(true));

map.put("data", replys);

return map;

}

private Map<String, Object> tIncept2Map(TIncept incept) {

Map map = new HashMap();

map.put("id", incept.getId());

map.put("notreplyDept", incept.getInceptDept());

return map;

}

private Map<String, Object> tReply2Map(TReply reply) {

Map map = new HashMap();

map.put("id", reply.getId());

map.put("replyPersonId", Integer.valueOf(reply.getReplyPersonId()));

map.put("replyPerson", reply.getReplyPerson());

map.put("replyDept", reply.getReplyDept());

if (reply.getReplyTime() != null)

map.put("replyTime", new SimpleDateFormat("yyyy-MM-dd")

.format(reply

.getReplyTime()));

else {

map.put("replyTime", reply.getReplyTime());

}

if ((reply.getReplyContent() != null) && (!"".equals(reply.getReplyContent())))

map.put("replyContent", reply.getReplyContent().replace("\n", "<br>").replace(" ", "&nbsp").replace("\t", "&nbsp&nbsp&nbsp&nbsp"));

else {

map.put("replyContent", reply.getReplyContent());

}

List flist = this.replyAttachmentService.findAttList(reply

.getId().intValue());

String files = "";

String fileid = "";

for (int i = 0; i < flist.size(); i++) {

files = files + ((TReplyFile)flist.get(i)).getFileName();

files = files + ",";

fileid = fileid + ((TReplyFile)flist.get(i)).getId();

fileid = fileid + ",";

}

if (files.length() > 1) {

files = files.substring(0, files.length() - 1);

}

if (fileid.length() > 1) {

fileid = fileid.substring(0, fileid.length() - 1);

}

map.put("files", files);

map.put("fileid", fileid);

map.put("fileList", flist);

return map;

}

@RequestMapping({"/docexchange/showBmPerson/list.do"})

@ResponseBody

public Map<String, Object> listbm(Integer sendId) {

Map map = new HashMap();

List bmpersons = new ArrayList();

if (sendId != null) {

List bmbean = this.bmService.getBySendId(sendId.intValue());

if (bmbean != null) {

for (int i = 0; i < bmbean.size(); i++)

bmpersons.add(bmPerson2Map((TBm)bmbean.get(i)));

}

}

map.put("rows", bmpersons);

return map;

}

@RequestMapping({"docexchange/send/listnotbm.do"})

public ModelAndView listnotbm(Integer limit, Integer cpage, String sendId)

{

if (limit == null) limit = Integer.valueOf(15);

if (cpage == null) {

cpage = new Integer(1);

}

int size = 0;

int gpage = (int)Math.ceil(size / limit.intValue());

if (cpage.intValue() <= 0) {

cpage = Integer.valueOf(1);

} else if (gpage == 0) {

cpage = Integer.valueOf(1);

gpage = 1;

} else if (cpage.intValue() > gpage) {

cpage = Integer.valueOf(gpage);

}

ModelAndView mav = new ModelAndView("/gwjh/hassent\_unbmry");

List bmpersons = new ArrayList();

if (sendId != null) {

List bmbean = this.bmService.getNotBmBySendId(Integer.parseInt(sendId));

if (bmbean != null) {

for (int i = 0; i < bmbean.size(); i++)

bmpersons.add(bmPerson12Map((TIncept)bmbean.get(i)));

}

}

mav.addObject("rows", bmpersons);

mav.addObject("sendId", sendId);

mav.addObject("cpage", cpage);

mav.addObject("gpage", Integer.valueOf(gpage));

mav.addObject("totalDeptNum", Integer.valueOf(bmpersons.size()));

return mav;

}

private Map<String, Object> bmPerson2Map(TBm bmbean)

{

Map map = new HashMap();

List bmpersonlist = this.bmPersonService.findlist(bmbean.getSendId().intValue(), bmbean.getDeptId().intValue());

Collections.sort(bmpersonlist, new Comparator() {

public int compare(TBmPerson arg0, TBmPerson arg1) {

return arg0.getPx().intValue() - arg1.getPx().intValue();

}

});

StringBuffer bmryBuffer = new StringBuffer();

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

for (int i = 0; i < bmpersonlist.size(); i++) {

if ((((TBmPerson)bmpersonlist.get(i)).getPersonname() != null) && (!((TBmPerson)bmpersonlist.get(i)).getPersonname().equals(""))) {

bmryBuffer.append(((TBmPerson)bmpersonlist.get(i)).getPersonname());

}

if ((((TBmPerson)bmpersonlist.get(i)).getGender() != null) && (!((TBmPerson)bmpersonlist.get(i)).getGender().equals(""))) {

bmryBuffer.append("," + ((TBmPerson)bmpersonlist.get(i)).getGender());

}

if ((((TBmPerson)bmpersonlist.get(i)).getDut() != null) && (!((TBmPerson)bmpersonlist.get(i)).getDut().equals(""))) {

bmryBuffer.append("," + ((TBmPerson)bmpersonlist.get(i)).getDut());

}

if (i < bmpersonlist.size() - 1) {

bmryBuffer.append("<br>");

}

}

}

String bmryJson = bmryBuffer.toString().replace("null", "");

map.put("uuid", bmbean.getUuid());

map.put("sendId", bmbean.getSendId());

map.put("deptname", bmbean.getDeptName());

if ((bmbean.getBmtime() == null) || (bmbean.getBmtime().equals("")))

map.put("bmtime1", "");

else {

map.put("bmtime1", new SimpleDateFormat("yyyy-MM-dd HH:mm:ss")

.format(bmbean

.getBmtime()));

}

map.put("bmNum", Integer.valueOf(bmpersonlist.size()));

map.put("bmry", bmryJson);

return map;

}

private Map<String, Object> bmPerson12Map(TIncept bmbean) {

Map map = new HashMap();

map.put("uuid", bmbean.getUuid());

map.put("notbmDept", bmbean.getInceptDept());

return map;

}

@RequestMapping({"docexchange/send/exportBm.do"})

public void exportBm(Integer sendId, HttpServletRequest request, HttpServletResponse response)

{

List bmbean = new ArrayList();

List notbmbean = new ArrayList();

List inceptlist = new ArrayList();

if (sendId != null) {

inceptlist = this.inceptService.findbySendId(sendId.intValue());

}

HSSFWorkbook wb = new HSSFWorkbook();

HSSFSheet sheet = wb.createSheet("报名表");

sheet.setColumnWidth(0, 2560);

sheet.setColumnWidth(1, 5120);

sheet.setColumnWidth(2, 2560);

sheet.setColumnWidth(3, 7680);

sheet.setColumnWidth(4, 10240);

sheet.setColumnWidth(5, 10240);

sheet.setColumnWidth(6, 10240);

HSSFRow row = sheet.createRow(0);

HSSFCellStyle style = wb.createCellStyle();

style.setWrapText(true);

style.setVerticalAlignment(1);

style.setBorderLeft(1);

style.setLeftBorderColor(8);

style.setBorderRight(1);

style.setRightBorderColor(8);

style.setBorderTop(1);

style.setTopBorderColor(8);

style.setBorderBottom(1);

style.setBottomBorderColor(8);

String[] excelHeader = { "序号", "姓名", "性别", "职务", "单位名称", "手机号码", "备注" };

style.setAlignment(2);

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

HSSFCell cell = row.createCell(i);

cell.setCellValue(excelHeader[i]);

cell.setCellStyle(style);

}

if (inceptlist.size() > 0) {

int m = 1;

int index = 1;

for (int i = 0; i < inceptlist.size(); i++) {

List bmpersonlist = this.bmPersonService.findExportList(sendId,

((TIncept)inceptlist

.get(i))

.getInceptDeptId());

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

for (int j = 0; j < bmpersonlist.size(); j++) {

row = sheet.createRow(m++);

row.createCell(0).setCellValue(index);

row.createCell(1).setCellValue(

((TBmPerson)bmpersonlist

.get(j))

.getPersonname());

row.createCell(2).setCellValue(

((TBmPerson)bmpersonlist

.get(j))

.getGender());

row.createCell(3).setCellValue(

((TBmPerson)bmpersonlist

.get(j))

.getDut());

row.createCell(4).setCellValue(

((TIncept)inceptlist

.get(i))

.getInceptDept());

row.createCell(5).setCellValue(

((TBmPerson)bmpersonlist

.get(j))

.getContacttel());

row.createCell(6).setCellValue(

((TBmPerson)bmpersonlist

.get(j))

.getComment());

row.getCell(0).setCellStyle(style);

row.getCell(1).setCellStyle(style);

row.getCell(2).setCellStyle(style);

row.getCell(3).setCellStyle(style);

row.getCell(4).setCellStyle(style);

row.getCell(5).setCellStyle(style);

row.getCell(6).setCellStyle(style);

index++;

}

}

if (bmpersonlist == null) {

row = sheet.createRow(m++);

row.createCell(0).setCellValue(index);

row.createCell(1).setCellValue("");

row.createCell(2).setCellValue("");

row.createCell(3).setCellValue("");

row.createCell(4).setCellValue(

((TIncept)inceptlist

.get(i))

.getInceptDept());

row.createCell(5).setCellValue("");

row.createCell(6).setCellValue("");

row.getCell(0).setCellStyle(style);

row.getCell(1).setCellStyle(style);

row.getCell(2).setCellStyle(style);

row.getCell(3).setCellStyle(style);

row.getCell(4).setCellStyle(style);

row.getCell(5).setCellStyle(style);

row.getCell(6).setCellStyle(style);

index++;

}

}

}

response.setContentType("application/vnd.ms-excel;charset=UTF-8");

response.setHeader("Content-disposition", "attachment;filename=bm.xls");

OutputStream ouputStream = null;

try {

ouputStream = response.getOutputStream();

wb.write(ouputStream);

ouputStream.flush();

ouputStream.close();

} catch (IOException e1) {

e1.printStackTrace();

}

}

@RequestMapping({"/docexchange/send/listfiles.do"})

@ResponseBody

public Map<String, Object> listfiles(Integer sendId) {

Map map = new HashMap();

if (sendId == null) {

map.put("success", Boolean.valueOf(false));

return map;

}

List files = this.sendService.findSendfiles(sendId.intValue());

List list = new ArrayList();

Iterator i = files.iterator();

while (i.hasNext()) {

list.add(sendFileToMap((TSendFile)i.next()));

}

files = null;

map.put("success", Boolean.valueOf(true));

map.put("data", list);

return map;

}

private Map<String, Object> sendFileToMap(TSendFile file) {

Map map = new HashMap();

map.put("id", file.getId());

map.put("fileName", file.getFileName());

map.put("fileType", file.getFileType());

map.put("fileSize", Integer.valueOf(file.getFileSize()));

map.put("fileAddress", file.getFileAddress());

return map;

}

@RequestMapping({"docexchange/send/cancel.do"})

@ResponseBody

public Map<String, Object> cancel(Integer inceptId) {

Map map = new HashMap();

this.sendService.cancelIncept(inceptId.intValue());

map.put("success", Boolean.valueOf(true));

return map;

}

@RequestMapping({"docexchange/send/listincept1.do"})

@ResponseBody

public Map<String, Object> listincept1(Integer start, Integer limit, Integer sendId) { Integer type = Integer.valueOf(1);

Map map = new HashMap();

if (sendId == null) {

map.put("success", Boolean.valueOf(false));

return map;

}

List list = this.sendService.findSendIncepts(start, limit, sendId.intValue(), type.intValue());

List incepts = new ArrayList();

Iterator i = list.iterator();

while (i.hasNext()) {

incepts.add(inceptToMap((TIncept)i.next()));

}

list = null;

map.put("success", Boolean.valueOf(true));

map.put("data", incepts);

return map; }

@RequestMapping({"docexchange/send/listincept.do"})

@ResponseBody

public Map<String, Object> listincept(Integer limit, Integer start, Integer sendId, Integer type) {

Map map = new HashMap();

if (sendId == null) {

map.put("success", Boolean.valueOf(false));

return map;

}

List list = this.sendService.findSendIncepts(start, limit, sendId.intValue(), type.intValue());

List incepts = new ArrayList();

Iterator i = list.iterator();

while (i.hasNext()) {

incepts.add(inceptToMap((TIncept)i.next()));

}

list = null;

Integer total = this.sendService.findSendCountIncepts(sendId, type);

map.put("success", Boolean.valueOf(true));

map.put("rows", incepts);

map.put("total", total);

return map;

}

private Map<String, Object> inceptToMap(TIncept incept) {

Map map = new HashMap();

map.put("id", incept.getId());

map.put("inceptDept", incept.getInceptDept());

map.put("inceptDeptId", Integer.valueOf(incept.getInceptDeptId()));

map.put("inceptor", incept.getInceptor());

map.put("inceptorId", Integer.valueOf(incept.getInceptorId()));

if (incept.getInceptTime() != null)

map.put("inceptTime", new SimpleDateFormat("yyyy-MM-dd HH:mm:ss")

.format(incept

.getInceptTime()));

else

map.put("inceptTime", incept.getInceptTime());

map.put("signReceive", Integer.valueOf(incept.getSignReceive()));

map.put("backFlag", Integer.valueOf(incept.getBackFlag()));

map.put("cancelFlag", Integer.valueOf(incept.getCancelFlag()));

map.put("replyFlag", Integer.valueOf(incept.getReplyFlag()));

map.put("sendFlag", Integer.valueOf(incept.getSendFlag()));

map.put("sendTitle", incept.gettSend().getSendTitle());

map.put("sendordept", incept.gettSend().getSendDept());

map.put("sendJjcd", incept.gettSend().getSendJjcd());

map.put("sendwh", incept.gettSend().getSendWh());

map.put("sendType", incept.gettSend().getSendType());

map.put("sendid", incept.gettSend().getId());

if (incept.gettSend().getSendTime() != null)

map.put("sendTime", new SimpleDateFormat("yyyy-MM-dd HH:mm:ss")

.format(incept

.gettSend().getSendTime()));

else {

map.put("sendTime", "");

}

if ((incept.getBackReason() != null) && (!"".equals(incept.getBackReason())))

map.put("backReason", incept.getBackReason().replace("\n", "<br>").replace(" ", "&nbsp").replace("\t", "&nbsp&nbsp&nbsp&nbsp"));

else {

map.put("backReason", incept.getBackReason());

}

if (incept.getBackTime() != null)

map.put("backTime", new SimpleDateFormat("yyyy-MM-dd HH:mm:ss")

.format(incept

.getBackTime()));

else {

map.put("backTime", incept.getBackTime());

}

if ((incept.getBatchStatus() != null) && (incept.getBatchStatus().intValue() == 0))

map.put("jobExecutionStatus", Integer.valueOf(0));

else {

map.put("jobExecutionStatus", Integer.valueOf(1));

}

TZzbm bmmodel = (TZzbm)this.zzbmService.get(incept.gettSend().getId());

if (bmmodel != null)

map.put("flag", bmmodel.getFlag());

return map;

}

@ResponseBody

public Map<String, Object> list(Integer limit, Integer start, String title, String startDate, String endDate, HttpSession session)

{

Map map = new HashMap();

List depts = (List)session.getAttribute("depts");

Integer deptId = ((TDept)depts.get(0)).getId();

Page page1 = this.sendService.findHasSend(start.intValue(), limit.intValue(), deptId.intValue(), title, startDate, endDate);

List tsendlist = page1.getItems();

List sends = new ArrayList();

Iterator i = tsendlist.iterator();

while (i.hasNext()) {

sends.add(TsendToMap((TSend)i.next()));

}

tsendlist = null;

map.put("total", Integer.valueOf(page1.getContext().getTotal()));

map.put("rows", sends);

return map;

}

@RequestMapping({"/gwjh/docexchange/send/list.do"})

public ModelAndView sendList(Integer limit, Integer cpage, String title, String startDate, String endDate, HttpSession session)

{

List depts = (List)session.getAttribute("depts");

Integer deptId = ((TDept)depts.get(0)).getId();

if (limit == null) limit = Integer.valueOf(15);

if (cpage == null) {

cpage = new Integer(1);

}

int size = 0;

Page page1 = this.sendService.findHasSend((cpage.intValue() - 1) \* limit.intValue(), limit.intValue(), deptId.intValue(), title, startDate, endDate);

size = page1.getContext().getTotal();

int gpage = (int)Math.ceil(size / limit.intValue());

if (cpage.intValue() <= 0) {

cpage = Integer.valueOf(1);

} else if (gpage == 0) {

cpage = Integer.valueOf(1);

gpage = 1;

} else if (cpage.intValue() > gpage) {

cpage = Integer.valueOf(gpage);

}

int start = (cpage.intValue() - 1) \* limit.intValue();

ModelAndView mav = new ModelAndView("/gwjh/hassent");

List tsendlist = page1.getItems();

List sends = new ArrayList();

Iterator i = tsendlist.iterator();

while (i.hasNext()) {

sends.add(TsendToMap((TSend)i.next()));

}

tsendlist = null;

mav.addObject("total", Integer.valueOf(size));

mav.addObject("rows", sends);

return mav;

}

private Map<String, Object> TsendToMap(TSend send) {

Map map = new HashMap();

map.put("id", send.getId());

map.put("sendor", send.getSendor());

map.put("sendorid", Integer.valueOf(send.getSendorId()));

map.put("sendDept", send.getSendDept());

map.put("sendDeptId", send.getSendDept());

map.put("sendTitle", send.getSendTitle());

map.put("sendTime", new SimpleDateFormat("yyyy-MM-dd").format(send

.getSendTime()));

map.put("sendTime1", new SimpleDateFormat("yyyy-MM-dd HH:mm:ss")

.format(send

.getSendTime()));

map.put("sendKeyWord", send.getSendKeyWord());

map.put("sendTitle1", send.getSendTitle1());

map.put("sendMj", send.getSendMj());

map.put("sendJjcd", send.getSendJjcd());

map.put("sendCopy", send.getSendCopy());

map.put("sendSignDept", send.getSendSignDept());

map.put("sendSigner", send.getSendSigner());

if (send.getSendSignTime() != null)

map.put("sendSignTime", new SimpleDateFormat("yyyy-MM-dd")

.format(send

.getSendSignTime()));

else {

map.put("sendSignTime", send.getSendSignTime());

}

map.put("sendScope", send.getSendScope());

map.put("sendType", send.getSendType());

map.put("sendWh", send.getSendWh());

int total = this.sendService.findTotalSend(send.getId().intValue());

int received = this.sendService.findReceived(send.getId().intValue());

int backed = this.sendService.findBacked(send.getId().intValue());

map.put("totleSend", Integer.valueOf(total));

map.put("received", Integer.valueOf(received));

map.put("notReceived", Integer.valueOf(backed));

map.put("sendLxr", send.getSendLxr());

map.put("sendLxdh", send.getSendLxdh());

map.put("sendPages", send.getSendPages());

return map;

}

@RequestMapping({"docexchange/send/dxcs.do"})

public void dxcs(String title, String ids, HttpServletResponse response, HttpSession session)

throws IllegalStateException, IOException

{

List depts = (List)session.getAttribute("depts");

if ((ids != null) && (ids != "")) {

String[] ids1 = ids.split(",");

List idlist = new ArrayList();

if ((ids != null) && (ids1.length > 0)) {

for (int i = 0; i < ids1.length; i++) {

Integer id = Integer.valueOf(Integer.parseInt(ids1[i]));

idlist.add(id);

}

this.sendService.dxcs(idlist, title, (TDept)depts.get(0));

}

}

}

@RequestMapping({"/docexchange/send/delsend.do"})

@ResponseBody

public Map<String, Object> delete(String sendIds) {

Map map = new HashMap();

String[] ids = sendIds.split(",");

if ((ids != null) && (ids.length > 0)) {

for (int j = 0; j < ids.length; j++) {

TSend send = (TSend)this.sendService.get(Integer.valueOf(Integer.parseInt(ids[j])));

if (send != null) {

List inceptList = this.inceptService.findbySendId(send.getId().intValue());

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

for (int k = 0; k < inceptList.size(); k++) {

TDept inceptDept = (TDept)this.deptService.get(Integer.valueOf(((TIncept)inceptList.get(k)).getInceptDeptId()));

if ((inceptDept.getAmqpFlag() != null) && (inceptDept.getAmqpFlag().intValue() == 2)) {

startJob(send.getUuid(), ((TIncept)inceptList.get(k)).getId(), null, "deleteGw");

}

}

}

}

}

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

int sendId = Integer.parseInt(ids[i]);

TAmqpTemp amqpTemp = new TAmqpTemp();

amqpTemp.setOperateType(Integer.valueOf(9));

amqpTemp.setCreateTime(new Date());

amqpTemp.setOperateUuid(Integer.valueOf(sendId));

amqpTemp.setFlag(Integer.valueOf(0));

this.amqpTempDao.save(amqpTemp);

}

}

map.put("success", Boolean.valueOf(true));

return map;

}

@RequestMapping({"/docexchange/send/exporthf.do"})

public void exportHf(Integer sendId, HttpServletRequest request, HttpServletResponse response)

{

List replylist = new ArrayList();

if (sendId != null) {

replylist = this.replyService.findBySendId(sendId.intValue());

}

HSSFWorkbook wb = new HSSFWorkbook();

HSSFSheet sheet = wb.createSheet("回复");

sheet.setColumnWidth(0, 5120);

sheet.setColumnWidth(1, 20480);

HSSFRow row = sheet.createRow(0);

HSSFCellStyle style = wb.createCellStyle();

style.setWrapText(true);

style.setVerticalAlignment(1);

style.setBorderLeft(1);

style.setLeftBorderColor(8);

style.setBorderRight(1);

style.setRightBorderColor(8);

style.setBorderTop(1);

style.setTopBorderColor(8);

style.setBorderBottom(1);

style.setBottomBorderColor(8);

String[] excelHeader = { "回复单位", "回复内容" };

style.setAlignment(2);

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

HSSFCell cell = row.createCell(i);

cell.setCellValue(excelHeader[i]);

cell.setCellStyle(style);

}

if (replylist.size() > 0) {

for (int i = 0; i < replylist.size(); i++) {

row = sheet.createRow(1 + i);

TReply reply = (TReply)replylist.get(i);

row.createCell(0).setCellValue(reply.getReplyDept());

row.createCell(1).setCellValue(reply.getReplyContent());

row.getCell(0).setCellStyle(style);

row.getCell(1).setCellStyle(style);

}

}

response.setContentType("application/vnd.ms-excel;charset=UTF-8");

response.setHeader("Content-disposition", "attachment;filename=reply.xls");

OutputStream ouputStream = null;

try {

ouputStream = response.getOutputStream();

wb.write(ouputStream);

ouputStream.flush();

ouputStream.close();

}

catch (IOException e1) {

e1.printStackTrace();

}

}

@RequestMapping({"/docexchange/send/cancelSend.do"})

@ResponseBody

public Map<String, Object> cancelGw(String inceptIds) { Map map = new HashMap();

if (inceptIds != null) {

String[] inceptStr = inceptIds.split(",");

if ((inceptStr != null) && (inceptStr.length > 0)) {

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

TIncept tIncept = (TIncept)this.inceptService.get(Integer.valueOf(inceptStr[i]));

if ((tIncept == null) ||

(tIncept.getSignReceive() != 0) || (tIncept.getCancelFlag() != 0)) continue;

TDept inceptDept = (TDept)this.deptService.get(Integer.valueOf(tIncept.getInceptDeptId()));

TSend tSend = (TSend)this.sendService.get(tIncept.gettSend().getId());

if (tSend != null) {

TDept sendDept = (TDept)this.deptService.get(Integer.valueOf(tSend.getSendDeptId()));

if ((inceptDept == null) || (sendDept == null) ||

(inceptDept.getActivemqQueueId() == null) || (sendDept.getActivemqQueueId() == null)) continue;

if ((inceptDept.getAmqpFlag() != null) && (inceptDept.getAmqpFlag().intValue() == 2)) {

startJob(tSend.getUuid(), tIncept.getId(), null, "cancel");

map.put("success", Boolean.valueOf(true));

}

if ((inceptDept.getAmqpFlag() != null) && (inceptDept.getAmqpFlag().intValue() == 0)) {

tIncept.setCancelFlag(1);

this.inceptService.update(tIncept);

map.put("success", Boolean.valueOf(true));

}

}

}

}

}

return map; }

@RequestMapping({"/docexchange/send/chongFaGw.do"})

@ResponseBody

public Map<String, Object> chongFaGw(String inceptIds) { Map map = new HashMap();

if ((inceptIds != null) && (inceptIds.length() > 0)) {

String[] inceptStrs = inceptIds.split(",");

if ((inceptStrs != null) && (inceptStrs.length > 0)) {

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

TIncept tblIncept = (TIncept)this.inceptService.get(Integer.valueOf(inceptStrs[i]));

if (tblIncept != null) {

TDept inceptDept = (TDept)this.deptService.get(Integer.valueOf(tblIncept.getInceptDeptId()));

TSend tSend = (TSend)this.sendService.get(tblIncept.gettSend().getId());

if (tSend != null) {

TDept sendDept = (TDept)this.deptService.get(Integer.valueOf(tSend.getSendDeptId()));

if ((inceptDept == null) || (sendDept == null) ||

(inceptDept.getActivemqQueueId() == null) || (sendDept.getActivemqQueueId() == null)) continue;

if ((inceptDept.getAmqpFlag() != null) && (inceptDept.getAmqpFlag().intValue() == 2)) {

startJob(tSend.getUuid(), tblIncept.getId(), null, "chongFa");

map.put("success", Boolean.valueOf(true));

}

if ((inceptDept.getAmqpFlag() != null) && (inceptDept.getAmqpFlag().intValue() == 0)) {

tblIncept.setCancelFlag(0);

this.inceptService.update(tblIncept);

map.put("success", Boolean.valueOf(true));

}

}

}

}

}

}

return map;

}

private String rename(String name)

{

Long now = Long.valueOf(Long.parseLong(new SimpleDateFormat("yyyyMMddHHmmss")

.format(new Date())));

Long random = Long.valueOf(()(Math.random() \* now.longValue()));

String fileName = now + "" + random;

fileName = fileName + name;

return fileName;

}

public void setSendService(ISendService sendService) {

this.sendService = sendService;

}

public ISendService getSendService() {

return this.sendService;

}

public String getDocexchangIp() {

return this.docexchangIp;

}

@Value("#{filepath['docexchange.ip']}")

public void setDocexchangIp(String docexchangIp) { this.docexchangIp = docexchangIp; }

@Value("#{filepath['filepath.root']}")

public void setRootPath(String rootPath) {

this.rootPath = rootPath;

}

public String getRootPath() {

return this.rootPath;

}

public IReplyService getReplyService() {

return this.replyService;

}

public void setReplyService(IReplyService replyService) {

this.replyService = replyService;

}

public IInceptService getInceptService() {

return this.inceptService;

}

public void setInceptService(IInceptService inceptService) {

this.inceptService = inceptService;

}

public IBmPersonService getBmPersonService() {

return this.bmPersonService;

}

public void setBmPersonService(IBmPersonService bmPersonService) {

this.bmPersonService = bmPersonService;

}

public IDeptService getDeptService() {

return this.deptService;

}

public void setDeptService(IDeptService deptService) {

this.deptService = deptService;

}

public IBmService getBmService() {

return this.bmService;

}

public void setBmService(IBmService bmService) {

this.bmService = bmService;

}

@InitBinder

public void initBinder(WebDataBinder binder) {

binder.registerCustomEditor(Date.class, new PropertyEditorSupport() {

public void setAsText(String value) {

try {

setValue(new SimpleDateFormat("yyyy-MM-dd").parse(value));

} catch (ParseException e) {

setValue(null);

}

}

public String getAsText() {

return new SimpleDateFormat("yyyy-MM-dd")

.format((Date)getValue());

}

});

}

@RequestMapping({"/attachment/upload.do"})

public void bpmAttachmentUpload(String gwjhUuid, HttpSession session, HttpServletRequest request, HttpServletResponse response) {

MultipartHttpServletRequest multipartRequest = (MultipartHttpServletRequest)request;

TUser user = (TUser)session.getAttribute("user");

CommonsMultipartFile file = (CommonsMultipartFile)multipartRequest

.getFile("Filedata");

String fileName1 = file.getOriginalFilename();

int fileSize = (int)file.getSize();

String[] types = fileName1.split("\\.");

int l = types.length - 1;

String fileType = types[l];

response.setContentType("text/html;charset=utf-8");

PrintWriter out = null;

try {

Calendar c = Calendar.getInstance();

int year = c.get(1);

int month = c.get(2) + 1;

int day = c.get(5);

String rootPath1 = File.separator + "file" + File.separator + "attachment" + File.separator + year + File.separator + month + File.separator + day + File.separator;

String realPath = this.rootPath + rootPath1;

CatalogUtil.createFolder(realPath);

String fileName = rename(fileName1);

CatalogUtil.createFolder(realPath);

File targetFile = new File(realPath, fileName);

if (!targetFile.exists()) {

targetFile.mkdirs();

}

file.transferTo(targetFile);

TSendFile attachment = new TSendFile();

attachment.setSendUuid(gwjhUuid);

attachment.setFileAddress(realPath + fileName);

attachment.setFileName(fileName1);

attachment.setFileType(fileType);

attachment.setFileSize(fileSize);

attachment.setUuid(gwjhUuid);

this.attachmentService.saveAttachment(attachment, request);

out = response.getWriter();

String json = "{fileName:'" + fileName1 + "',uuid:'" + gwjhUuid + "',fileSize:'" + fileSize + "'}";

out.println(json);

out.flush();

out.close();

} catch (Exception e) {

System.out.println("保存文件失败 !");

e.printStackTrace();

}

}

@RequestMapping({"/attachment/AttachmentList.do"})

@ResponseBody

public Map<String, Object> AttachmentList(String sendUuid, Integer limit, Integer start) {

Map map = new HashMap();

List attachmentlist = this.attachmentService.findAttachmentList(sendUuid, start, limit);

int total = this.attachmentService.findTotal(sendUuid);

List list = new ArrayList();

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

for (int i = 0; i < attachmentlist.size(); i++)

{

Map m = new HashMap();

m.put("uuid", ((TSendFile)attachmentlist.get(i)).getUuid());

m.put("filename", ((TSendFile)attachmentlist.get(i)).getFileName());

m.put("id", ((TSendFile)attachmentlist.get(i)).getId());

list.add(m);

}

}

map.put("total", Integer.valueOf(total));

map.put("rows", list);

return map;

}

@RequestMapping({"/attachment/deleteAttachment.do"})

public void deleteAttachment(String uuids, HttpServletResponse response, HttpServletRequest request, HttpSession session) {

this.attachmentService.deleteAttachment(uuids);

}

@RequestMapping({"/attachment/AttList.do"})

@ResponseBody

public Map<String, Object> AttList(Integer sendId, Integer limit, Integer start) {

Map map = new HashMap();

List attachmentlist = this.attachmentService.findAttList(sendId, start, limit);

int total = this.attachmentService.findTotal1(Integer.valueOf(sendId.intValue()).intValue());

System.out.println("---------------------------------------------------------total=" + total);

List list = new ArrayList();

for (int i = 0; i < attachmentlist.size(); i++) {

Map m = new HashMap();

TSend send = this.sendService.findByUuid(((TSendFile)attachmentlist.get(i)).getUuid());

m.put("uuid", ((TSendFile)attachmentlist.get(i)).getUuid());

m.put("filename", ((TSendFile)attachmentlist.get(i)).getFileName());

m.put("id", ((TSendFile)attachmentlist.get(i)).getId());

list.add(m);

}

map.put("total", Integer.valueOf(total));

map.put("rows", list);

return map;

}

@RequestMapping({"/attachment/downloadAttachment.do"})

public void downloadAttachment(Integer uuid, HttpServletRequest request, HttpServletResponse response, HttpSession session)

{

response.setCharacterEncoding("utf-8");

response.setContentType("application/x-msdownload;");

TSendFile attachment = this.attachmentService.Attachments(uuid);

String fileName = attachment.getFileName();

Integer fileSize = Integer.valueOf(attachment.getFileSize());

String filePath = attachment.getFileAddress();

InputStream inputStream = null;

OutputStream os = null;

this.attachmentService.downloadAttachment(request);

try

{

response.setHeader("Content-Disposition", "attachment;fileName=" +

new String(fileName

.getBytes("GBK"),

"ISO8859-1"));

response.setHeader("Content-Length", String.valueOf(fileSize));

inputStream = new FileInputStream(filePath);

os = response.getOutputStream();

byte[] b = new byte[1024];

int length;

while ((length = inputStream.read(b)) > 0)

os.write(b, 0, length);

}

catch (UnsupportedEncodingException e)

{

e1.printStackTrace();

} catch (FileNotFoundException e) {

e.printStackTrace();

} catch (IOException e) {

e.printStackTrace();

} finally {

if (inputStream != null)

try {

inputStream.close();

}

catch (IOException e) {

e.printStackTrace();

}

if (os != null)

try {

os.close();

}

catch (IOException e) {

e.printStackTrace();

}

}

}

@RequestMapping({"/att/downloadAttachment.do"})

public void downloadAtt(String sendId, HttpServletRequest request, HttpServletResponse response, HttpSession session)

{

response.setCharacterEncoding("utf-8");

response.setContentType("application/x-msdownload;");

TSendFile attachment = this.attachmentService.Att(

Integer.valueOf(sendId).intValue());

String fileName = attachment.getFileName();

Integer fileSize = Integer.valueOf(attachment.getFileSize());

String filePath = attachment.getFileAddress();

InputStream inputStream = null;

OutputStream os = null;

this.attachmentService.downloadAttachment(request);

try

{

File file = new File(filePath);

response.setHeader("Content-Disposition", "attachment;fileName=" +

new String(fileName

.getBytes("GBK"),

"ISO8859-1"));

response.setHeader("Content-Length", String.valueOf(file.length()));

inputStream = new FileInputStream(file);

os = response.getOutputStream();

byte[] b = new byte[1024];

int length;

while ((length = inputStream.read(b)) > 0) {

os.write(b, 0, length);

}

}

catch (UnsupportedEncodingException e)

{

e1.printStackTrace();

} catch (FileNotFoundException e) {

e.printStackTrace();

} catch (IOException e) {

e.printStackTrace();

} finally {

if (inputStream != null)

try {

inputStream.close();

}

catch (IOException e) {

e.printStackTrace();

}

if (os != null)

try {

os.close();

}

catch (IOException e) {

e.printStackTrace();

}

}

}

private TSendFile copyFile(MultipartFile file) throws IllegalStateException, IOException

{

if (file.getSize() == 0L)

return null;

Calendar c = Calendar.getInstance();

int year = c.get(1);

int month = c.get(2) + 1;

int day = c.get(5);

String rootPath1 = File.separator + "file" + File.separator + "attachment" + File.separator + year + File.separator + month + File.separator + day + File.separator;

String realPath = this.rootPath + rootPath1;

CatalogUtil.createFolder(realPath);

String fileName1 = file.getOriginalFilename();

String fileName = rename(fileName1);

File targetFile = new File(realPath, fileName);

if (!targetFile.exists()) {

targetFile.mkdirs();

}

file.transferTo(targetFile);

TSendFile sendFile = new TSendFile();

sendFile.setFileName(file.getOriginalFilename());

sendFile.setFileType(file.getContentType());

sendFile.setFileAddress(targetFile.getPath());

sendFile.setFileSize((int)file.getSize());

return sendFile; }

@RequestMapping({"/docexchange/send/failedResend.do"})

@ResponseBody

public Map<String, Object> failedResend(Integer inceptId) { Map map = new HashMap();

TIncept tIncept = (TIncept)this.inceptService.get(inceptId);

if (tIncept != null) {

TSend tSend = (TSend)this.sendService.get(tIncept.gettSend().getId());

if ((tSend != null) &&

(tSend.getUuid() != null)) {

startJob(tSend.getUuid(), tIncept.getId(), null, "failedResend");

}

map.put("sendId", tIncept.gettSend().getId());

}

map.put("success", Boolean.valueOf(true));

return map;

}

public void startJob(String sendUuid, Integer inceptId, String bmUuid, String type)

{

JobParametersBuilder builder = new JobParametersBuilder();

builder.addString("sUuid", sendUuid);

builder.addString("inceptId", inceptId.toString());

builder.addString("bmUuid", bmUuid);

builder.addString("type", type);

try

{

this.jobLauncher.run(this.job, builder.toJobParameters());

}

catch (JobExecutionAlreadyRunningException e) {

e.printStackTrace();

}

catch (JobRestartException e) {

e.printStackTrace();

}

catch (JobInstanceAlreadyCompleteException e) {

e.printStackTrace();

}

catch (JobParametersInvalidException e) {

e.printStackTrace();

}

}

}

GwjhAttachmentcontroller.java

package com.jiexunnet.docexchange.web.controller;

import com.jiexunnet.docexchange.model.TReplyFile;

import com.jiexunnet.docexchange.model.TSendFile;

import com.jiexunnet.docexchange.service.GwjhAttachmentService;

import com.jiexunnet.docexchange.service.ReplyAttachmentService;

import com.jiexunnet.common.web.support.editor.DateTimeEditor;

import com.jiexunnet.usm.service.IUserService;

import java.io.File;

import java.io.FileInputStream;

import java.io.FileNotFoundException;

import java.io.IOException;

import java.io.InputStream;

import java.io.UnsupportedEncodingException;

import java.text.SimpleDateFormat;

import java.util.ArrayList;

import java.util.Date;

import java.util.HashMap;

import java.util.List;

import java.util.Map;

import javax.servlet.ServletOutputStream;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

import javax.servlet.http.HttpSession;

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

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

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

import org.springframework.stereotype.Controller;

import org.springframework.web.bind.WebDataBinder;

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

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

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

@Controller

public class GwjhAttachmentController

{

@Autowired

@Qualifier("GwjhAttachmentService")

private GwjhAttachmentService attachmentService;

@Autowired

@Qualifier("IUserService")

private IUserService userService;

@Autowired

@Qualifier("ReplyAttachmentService")

private ReplyAttachmentService replyAttachmentService;

private String rootPath;

@RequestMapping({"/oa/gwjh/attachment/downloadAttachment.do"})

public void downloadAttachment(String uuid, String type, HttpServletRequest request, HttpServletResponse response, HttpSession session)

{

response.reset();

response.setCharacterEncoding("utf-8");

response.setContentType("application/x-msdownload;");

String fileName = "";

String filePath = "";

if (type.endsWith("send")) {

TSendFile attachment = (TSendFile)this.attachmentService.get(Integer.valueOf(Integer.parseInt(uuid)));

fileName = attachment.getFileName();

filePath = attachment.getFileAddress();

}

else if (type.endsWith("reply")) {

TReplyFile attachment = (TReplyFile)this.replyAttachmentService.get(Integer.valueOf(Integer.parseInt(uuid)));

fileName = attachment.getFileName();

filePath = attachment.getFileAddress();

}

InputStream inputStream = null;

ServletOutputStream os = null;

this.attachmentService.downloadAttachment(request);

try

{

File file = new File(filePath);

response.setHeader("Content-Disposition", "attachment;fileName=" +

new String(fileName

.getBytes("GBK"),

"ISO8859-1"));

response.setHeader("Content-Length", String.valueOf(file.length()));

inputStream = new FileInputStream(file);

os = response.getOutputStream();

byte[] b = new byte[1024];

int length;

while ((length = inputStream.read(b)) > 0) {

os.write(b, 0, length);

}

os.flush();

}

catch (UnsupportedEncodingException e) {

e1.printStackTrace();

} catch (FileNotFoundException e) {

e.printStackTrace();

} catch (IOException e) {

e.printStackTrace();

} finally {

if (inputStream != null)

try {

inputStream.close();

}

catch (IOException e) {

e.printStackTrace();

}

if (os != null)

try {

os.close();

}

catch (IOException e) {

e.printStackTrace();

}

}

}

@RequestMapping({"/attachment/AttachmentList1.do"})

@ResponseBody

public Map<String, Object> AttachmentList(String sendUuid, Integer limit, Integer start)

{

Map map = new HashMap();

List attachmentlist = this.attachmentService.findAttachmentList(sendUuid, start, limit);

int total = this.attachmentService.findTotal(sendUuid);

List list = new ArrayList();

for (int i = 0; i < attachmentlist.size(); i++) {

Map m = new HashMap();

m.put("uuid", ((TSendFile)attachmentlist.get(i)).getUuid());

m.put("filename", ((TSendFile)attachmentlist.get(i)).getFileName());

list.add(m);

}

map.put("total", Integer.valueOf(total));

map.put("rows", list);

return map;

}

public String timeZhuanHuan(Date createDate)

{

SimpleDateFormat formater = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss");

return formater.format(createDate);

}

@InitBinder

public void initBinder(WebDataBinder binder) {

binder.registerCustomEditor(Date.class, new DateTimeEditor());

}

public GwjhAttachmentService getAttachmentService()

{

return this.attachmentService;

}

public void setAttachmentService(GwjhAttachmentService attachmentService) {

this.attachmentService = attachmentService;

}

@Value("#{filepath['filepath.root']}")

public void setRootPath(String rootPath) {

this.rootPath = rootPath;

}

public String getRootPath() {

return this.rootPath;

}

public IUserService getUserService() {

return this.userService;

}

public void setUserService(IUserService userService) {

this.userService = userService;

}

}

**\login.jsp**

<%@ page language="java" import="java.util.\*" pageEncoding="utf-8"%>

<%

String path = request.getContextPath();

String basePath = request.getScheme()+"://"+request.getServerName()+":"+request.getServerPort()+path+"/";

%>

<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN" "http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">

<html xmlns="http://www.w3.org/1999/xhtml">

<head>

<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />

<title>科展信达文件交换平台V2.0</title>

<link rel="stylesheet" type="text/css" href="css/login.css"/>

<!--图片透明-->

<!--[if IE 6]>

<script type="text/javascript" src="js/DD\_belatedPNG.js" ></script>

<script type="text/javascript">DD\_belatedPNG.fix('.login,.login\_01 img,');</script>

<![endif]-->

<!--此代码是左右折叠的JS代码-->

<script type="text/javascript">

function msg() {

var message = document.getElementById("message").value;

if(message.length > 0){

alert(message);

}

}

function switchSysBar(){

if (switchPoint.innerText==3){

switchPoint.innerText=4;

document.all("www\_zzjs\_net").style.display="none";

}

else{

switchPoint.innerText=3;

document.all("www\_zzjs\_net").style.display="";

}

}

</script>

</head>

<body onload="msg()">

<img src="/images/bg.jpg" class="img" style="height:100%; width:100%;">

<form action="/login.do" method="post">

<div class="login">

<div class="login\_01 pt45"><img src="images/logo.png" /></div>

<div class="login\_02">

<div class="login\_021 fLeft">

<div class="name fLeft"><img src="images/login\_03.jpg" /></div>

<input type="text" id="loginName" name="loginName" value="" width="100" style="width:160px; height:36px; line-height:36px; margin:0; padding:0; border:none; margin-top:1px; margin-left:10px; overflow:hidden;"/>

</div>

<div class="login\_021 fLeft" style="margin-left:15px; overflow:hidden;">

<div class="name fLeft"><img src="images/login\_04.jpg" /></div>

<input type="password" id="password" name="password" value="" style="width:160px; height:36px; line-height:36px; margin:0; padding:0; border:none; margin-top:1px; margin-left:10px; overflow:hidden;" />

<input type="text" class="name\_text" />

<input id="message" name="message" value="${message }" />

</div>

<input name="image" type="image" id="img" src="images/login\_01.png" style="float:left;margin-left:10px;width:90px;height:37px;" />

<div class="fRight"></div>

<div class="clearboth"></div>

</div>

</div>

</form>

</body>

</html>

**\index.jsp**

<%@ page language="java" contentType="text/html; charset=UTF-8" pageEncoding="UTF-8"%>

<%

String path = request.getContextPath();

String basePath = request.getScheme() + "://"

+ request.getServerName() + ":" + request.getServerPort()

+ path + "/";

%>

<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core"%>

<html>

<head>

<meta http-equiv="Content-Type" content="text/html; charset=UTF-8">

<title>科展信达文件交换平台V2.0</title>

<link href="<c:url value='/js/extjs/resources/css/ext-all.css'/>"

rel="stylesheet" type="text/css"></link>

<style>

.style3 {

cursor:pointer

}

.x-btn-text-icon .x-btn-icon-small-left .x-btn-text{

background-position: 0 center;/\*默认\*/

background-repeat: no-repeat;/\*默认\*/

padding-left:18px;/\*默认\*/

height:16px;/\*默认\*/

font-size:15px;/\*字体大小设置\*/

}

</style>

<script type="text/javascript" src="<c:url value='/js/extjs/ext-all.gzjs'/>"></script>

<script type="text/javascript" src="<c:url value='/js/docexchange/app.js'/>"></script>

</head>

<body>

<BGSOUND ID="bkgroundID" LOOP=0 VOLUME="-600" SRC="" mce\_SRC="/images/ti.mp3">

</body>

<script type="text/javascript">

Ext.onReady(function(){

loadNull();

});

function loadNull(){

setTimeout(loadMsgref, 1000 \* 20);

}

function loadMsgref(){

if(Ext.getCmp('noticewin')!=null){

Ext.getCmp('noticewin').close();

}

Ext.Ajax.request({

url:'/docexchange/incept/tixing.do',

method:'post',

success:function(o){

if(o.responseText>0){

var win= Ext.create('Ext.window.Window', {

title: '科展信达文件交换平台提醒',

height: 100,

width: 200,

layout: 'fit',

id:'noticewin',

renderTo: Ext.getBody(),

items: {

xtype: 'grid',

border: false,

store : 'Tixings',

columns: {items : [

{header:' ', sortable : false,dataIndex:'num', align:'center',width : 150,columnWidth:150,renderer:function(value){

return '<a href=\"javascript:void(0);\" onclick=\"openWindow();\">'+'共有'+value+'条公文</a>';

}}

]}

},

listeners : {

show :{

fn : function(it,eOpts){

var grid= it.down('grid');

var st = grid.getStore();

st.load();

}

}

}

});

win.setPosition( Ext.getBody().getWidth()-200, Ext.getBody().getHeight()-100);

win.show();

document.getElementById('bkgroundID').src='/images/ti.mp3';

}

}

});

setTimeout(loadMsgref, 1000 \* 30);

}

function openWindow(){

var tabs = Ext.getCmp('docexchange-main');

var id = 'unreceived-tab';

var xtype = 'unreceivedtab';

var tab = tabs.getComponent(id);

if(tab == null){

tabs.add({xtype:xtype, id:id}).show();

}

else{

tabs.setActiveTab(tab);

}

}

</script>

</html>

**\hf.jsp**

<%@ page language="java" import="java.util.\*,com.jiexunnet.usm.model.TUser,com.jiexunnet.usm.model.\*" pageEncoding="utf-8"%>

<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core"%>

<%

String path = request.getContextPath();

String basePath = request.getScheme() + "://"

+ request.getServerName() + ":" + request.getServerPort()

+ path + "/";

List<TDept> depts = (List<TDept>) session.getAttribute("depts");

String deptName = depts.get(0).getDeptName();

String uuid = UUID.randomUUID().toString();

%>

<!DOCTYPE HTML>

<html>

<head>

<c:import url="/oa/index/oaIncludeJs.do" />

<link rel="stylesheet" type="text/css" href="/css/Form.css" />

<link rel="stylesheet" type="text/css" href="/css/detail.css" />

<style>

#HfForm .null,#HfForm .error,#HfForm .focus {

color: red;

line-height: 22px;

line-height: 20px;

}

</style>

</head>

<body >

<form method="post" id="HfForm" action="">

<div class="address">

<div class="address\_01 fLeft">

当前位置：

<a href="/user/shouye.do">公文接收箱</a> >

<a href="/gwjh/received.do">已收列表</a> >

<a href="#" onClick="history.go(-1)">查看文件</a> >

回复

</div>

</div>

<div class="table\_outside">

<div class="jy\_detail\_title">

<div class="jy\_btn btn\_right"><a href="javascript:void(0);" class="bg\_blue" onClick="return1()">返回</a></div>

<div class="jy\_btn btn\_left"><a href="javascript:void(0);" class="bg\_orange" id="hf">回复</a></div>

</div>

<div class="table\_box" id="table\_box">

<div class="table\_height" id="table\_height">

<div class="jy\_detail\_table">

<input type="hidden" id="atcuuid" name="atcuuid" value="<%=uuid%>" />

<input type="hidden" id="inceptId" name="inceptId" value="${inceptId }"/>

<table>

<tr>

<td class="td\_left">回执人:</td>

<td>

<input type="text" id="replyPerson" readonly="readonly" name="replyPerson" value="<%=deptName%>"/>

</td>

<td class="td\_left">回执单位:</td>

<td>

<input type="text" id="replyDept" name="replyDept" readonly="readonly" value="<%=deptName%>" />

</td>

</tr>

<tr>

<td class="td\_left">内容:</td>

<td colspan="3">

<textarea class="xtd\_area" id="replyContent" name="replyContent"></textarea>

</td>

</tr>

<tr>

<td class="td\_left">附件:</td>

<td colspan="3">

<div id="attachementDiv"></div>

</td>

</tr>

</table>

</div>

</div>

</div>

</div>

</form>

</body>

<script type="text/javascript">

$("#attachementDiv").omExtendAttachement({

uploadurl : '/Replyattachment/upload.do', //上传路径

gridurl : '/Replyattachment/AttachmentList.do?uuid=<%=uuid%>', //数据列表路径

delurl : '/Replyattachment/deleteAttachment.do', //删除url

uploadActionData : {'uuid' : '<%=uuid%>'}, //上传时附加的参数

width :680, //列表宽度

height : 230, //列表高度

limit : 5, //grid数据条数

colModel : [{

header:'文件名称',

name:'filename',

width:602,

align : 'center',

renderer : function(colValue, rowData, rowIndex) {

return "<a href='/Replyattachment/downloadAttachment.do?id="+rowData.id+"'>"+colValue+"</a>";

}

}] //grid列定义

});

$('#hf').click(function(){

if($('#replyContent').val() == "" ||$.trim($("#replyContent").val())==""){

alert("回复内容不能为空，请检查是否填写！！！");

}else{

$('#HfForm').attr('action','/docexchange/reply/save.do');

$('#HfForm').submit();

}

});

</script>

</html>

**Jq.jsp**

<%@ page language="java" import="java.util.\*,com.jiexunnet.usm.model.TUser,com.jiexunnet.usm.model.\*" pageEncoding="utf-8"%>

<%@ taglib prefix="c" uri="http://java.sun.com/jsp/jstl/core"%>

<%

String path = request.getContextPath();

String basePath = request.getScheme() + "://"

+ request.getServerName() + ":" + request.getServerPort()

+ path + "/";

List<TDept> depts = (List<TDept>) session.getAttribute("depts");

String deptName = depts.get(0).getDeptName();

String uuid = UUID.randomUUID().toString();

%>

<!DOCTYPE HTML>

<html>

<head>

<c:import url="/oa/index/oaIncludeJs.do" />

<link rel="stylesheet" type="text/css" href="/css/Form.css" />

<link rel="stylesheet" type="text/css" href="/css/detail.css" />

<style>

#JqForm .null,#JqForm .error,#JqForm .focus {

color: red;

line-height: 22px;

line-height: 20px;

}

</style>

</head>

<body>

<form method="post" id="JqForm" action="">

<div class="address">

<div class="address\_01 fLeft">

当前位置：

<a href="/user/shouye.do">公文接收箱</a> >

<a href="/gwjh/received.do">已收列表</a> >

<a href="#" onClick="history.go(-1)">查看文件</a> >

拒签

</div>

</div>

<div class="table\_outside">

<div class="jy\_detail\_title">

<div class="jy\_btn btn\_right"><a href="javascript:void(0);" class="bg\_blue" onClick="return1()">返回</a></div>

<div class="jy\_btn btn\_left"><a href="javascript:void(0);" class="bg\_orange" id="Jq">拒签</a></div>

</div>

<div class="table\_box" id="table\_box">

<div class="table\_height" id="table\_height">

<div class="jy\_detail\_table">

<table>

<tr>

<td class="td\_left">拒签人:</td>

<td>

<input type="text" id="replyPerson" readonly="readonly" name="replyPerson" value="<%=deptName%>" />

</td>

<td class="td\_left">拒签单位:</td>

<td>

<input type="text" id="replyDept" name="replyDept" readonly="readonly" value="<%=deptName%>" />

</td>

</tr>

<tr>

<td class="td\_left">拒签原因:</td>

<td colspan="3">

<textarea id="backReason" name="backReason"></textarea>

</td>

</tr>

</table>

</div>

</div>

</div>

</div>

</form>

</body>

</html>

<script type="text/javascript">

$('#Jq').click(function(){

if($('#backReason').val() == "" || null == $('#backReason').val() || $('#backReason').val().length > 200||$.trim($("#backReason").val())==""){

alert("拒签原因为必填项且不能超过200字，请核对！");

}else{

$('#JqForm').attr('action','/docexchange/incept/updateBackFlag.do?inceptId=${inceptid}');

$('#JqForm').submit();

alert("发送成功");

}

});

// 在鼠标聚焦到输入框的时候更改样式

$('#JqForm input, #JqForm textarea').bind('focus',function(){

var item = $(this);

item.removeClass('highlight2').addClass('highlight1');

var name = item.attr('name');

item.parent().find('#' + name + '\_error').attr('class','focus').html(focusMsg[name]);

}).bind('blur',function(){

var item = $(this);

item.removeClass('highlight1');

if(item.hasClass('highlight2')){

return;

}

var name = item.attr('name');

item.parent().find('#' + name + '\_error').attr('class','null').html('');

});

function return1(){

history.back();

}

</script>