1.考勤统计项目

项目描述:

统计公司当月以及上个月的考勤记录，依照给定的Excel模板生成统计Excel文件，并自动将生成的文件通过邮件群发给公司的副总。

项目需求：

(1)按部门分成两组(同时进行按部门排序)，生成两个文件。

(2)迟到在15分钟内的，有三次豁免机会，不用统计显示。

(3)迟到+部门调整忘打卡次数累计达5次或6次（即豁免次数）时，需显示在“达到上限”一栏内。

(4)如有请假情况，当日的缺勤记录将不会显示，而是显示在对应的请假类别内。

(5)每个考勤类别内，部门领导人需置顶并标红，其他数据按部门排序

工作流程内容：

(1)了解考勤记录在数据库内的存储关系，有用户信息表，打卡记录表，打卡时间记录表，请假记录表，代码字典表。

(2)确定数据查询逻辑：先从打卡记录表内获取相应时间段内的缺勤记录（status = 0），在从打卡时间记录表内获取与缺勤记录相应的打卡时间记录，然后再查询请假记录表，看是否有对应的请假记录。

(3)建立基本程序框架，建立一个Attendance类来统计缺勤记录和相应的缺勤时间记录。建立一个Affair类来统计请假信息。

(4)数据读取逻辑：多表查询获取指定时间段内的所有考勤记录以及相应的打开时间信息和用户信息存入Attendance对象集合，并按日期分类存放Attendance对象集合（Map）。然后，根据日期获取当天相应的所有请假记录存入Affair对象集合，并进行分类操作。

分类操作：将该集合与当日的Attendance对象集合进行比对，如果能对应，则将该请假时间存入该Attendance对象内，并将该对象映射到存储Map上（根据要生成的文件内部分类关系建立的存储Map<统计类别，Map<userid, Attendance对象集合>）。

分类操作难点：如果当日上下班皆未打卡，但只有下午有请假。此时需要过滤掉下午的缺勤记录，只显示上午的缺勤记录。

解决方案：

//分类管理

public void sort(String key, List<Attendance> value) throws Exception {

logger.info("start to sort the affair type...");

List<Affair> list = dao.getAffairList(key);

List<Attendance> attends = new LinkedList<Attendance>();

attends.addAll(value);

for (Affair affair : list) {

for (Attendance attendance : attends) {

if(attendance.getId().equals(affair.getUserId())){

Attendance attend = new Attendance(attendance.getId(), attendance.getCode(), attendance.getName(), attendance.getDate(), attendance.getInStatus(), attendance.getOutStatus(), attendance.getRoleId(), attendance.getCheckInTime(), attendance.getCheckOutTime(), attendance.getAffair(), attendance.getDepartmentId());

attend.setAffair(affair);

int type = Integer.parseInt(affair.getType());

addMap(type, attend);

if(attendance.getInStatus() == 0 && attendance.getOutStatus() == 0){

if(affair.getEndTime() == null){

if("上午".equals(affair.getStartTime())){

attendance.setInStatus(1);

}

if("下午".equals(affair.getStartTime())){

attendance.setOutStatus(1);

}

}else{

attendance.setAffair(affair);

}

}else{

attendance.setAffair(affair);

}

}

}

}

for (Attendance attend2 : attends) {

if(attend2.getAffair() == null){

if(attend2.getInStatus() == 0 || attend2.getOutStatus() == 0){

addMap(0,attend2);

}

}

}

之前采用的是父遍历Attendance对象集合，然后再嵌套子遍历Affair对象集合来进行userid比对，发现因为Attendance对象集合只遍历一遍，但可能该用户当天有两条请假记录，所有进行父子反转修改。这里new Attendance是为了让setAffair操作不影响原始对象，将其分离开来。通过setInstatus，setOutStatus以及setAffair，来解决上下午缺勤记录的分离。

(5)利用分类后的存储Map进行文件生成。首先读取配置文件的部门分类信息，进行两次文件生成操作，每次操作只获取相应部门内人员的缺勤记录。以下三个类为了解决需求，有特殊处理。

迟到：

if(freeCount < MAX\_LATE\_COUNT){

if(checkLateTime(attend.getCheckInTime())){

//15分钟以内的三次不显示，但是计入上限限制

add(AttendanceType.UPLIMIT.ordinal(),attend);

freeCount++;

continue;

}

}

忘打卡:

//计入豁免上限限制

add(AttendanceType.UPLIMIT.ordinal(),attend);

接近上限：

List<Attendance> value = entry.getValue();

if(entry.getValue().size() == 5 || entry.getValue().size() == 6){content}

部门领导人标红：

if("2".equals(attend.getRoleId().trim())){

cell.setCellStyle(redStyle);

}else{

cell.setCellStyle(cellStyle);

}

部门领导人置顶+部门排序：

private void resort(Map<String, List<Attendance>> map, LinkedList<Attendance> writeList) {

Map<String,List<Attendance>> depMap = new HashMap<String,List<Attendance>>();

List<Attendance> newList = null;

for(Entry<String,List<Attendance>> entry : map.entrySet()){

List<Attendance> value = entry.getValue();

for (Attendance attend : value) {

if(!deptIds.contains(attend.getDepartmentId())){

continue;

}

if("2".equals(attend.getRoleId())){

writeList.add(attend);

}else{

if(depMap.containsKey(attend.getDepartmentId())){

depMap.get(attend.getDepartmentId()).add(attend);

}else{

newList = new LinkedList<Attendance>();

newList.add(attend);

depMap.put(attend.getDepartmentId(), newList);

}

}

}

}

for (String deptId : deptIds) {

if(null != depMap.get(deptId)){

writeList.addAll(depMap.get(deptId));

}

}

}

(6)将利用POI技术生产的Excel文件，以邮件附件形式发送给领导。利用java.mailAPI来开发邮件发送程序。

心得：

1）解析主机服务器域名需要配置DNS。

2）要实现邮件发送，要确认访问权限以及端口状态，以及是否支持SSL。还需确认接收方邮箱是否有访问权限限制，以及smtp权限。

3）多附件发送时，正文内容和附件的顺序在某些邮件服务器上会有不同，例如：如果将正文内容作为最后一个 MimeBodyPart，有的邮件服务器会将其作为一个附件生成。

4）MimeMessage设置邮件内容，Session用来进行邮件收发设置。

package com.mail;

import java.io.UnsupportedEncodingException;

import java.util.Date;

import java.util.List;

import java.util.Properties;

import javax.activation.DataHandler;

import javax.activation.FileDataSource;

import javax.mail.Authenticator;

import javax.mail.MessagingException;

import javax.mail.Multipart;

import javax.mail.PasswordAuthentication;

import javax.mail.Session;

import javax.mail.Transport;

import javax.mail.internet.AddressException;

import javax.mail.internet.InternetAddress;

import javax.mail.internet.MimeBodyPart;

import javax.mail.internet.MimeMessage;

import javax.mail.internet.MimeMessage.RecipientType;

import javax.mail.internet.MimeMultipart;

import javax.mail.internet.MimeUtility;

import org.apache.log4j.Logger;

public class MailSender {

Logger logger = Logger.getLogger(MailSender.class);

private String host;

private String port;

private String username;

private String password;

private String from;

private String nick;

public MailSender() {

// TODO Auto-generated constructor stub

}

public MailSender(String host, String port, String username, String password, String from,

String nick) {

super();

this.host = host;

this.port = port;

this.username = username;

this.password = password;

this.from = from;

this.nick = nick;

}

public boolean sendMail(String to, String subject, String body,

List<String> filepaths) throws UnsupportedEncodingException, AddressException, MessagingException{

logger.info("host:"+host);

logger.info("port:"+port);

logger.info("username:"+username);

logger.info("password:"+password);

logger.info("from:"+from);

logger.info("nick:"+nick);

logger.info("to:"+to);

if(body == null){

body = "";

}

if(subject == null){

subject = "";

}

Properties props = new Properties();

// 创建信件服务器

props.put("mail.smtp.host", host);

props.put("mail.smtp.port", port);

props.put("mail.smtp.auth", true);

props.put("mail.smtp.ssl.enable", false);

props.put("mail.smtp.username", username);

props.put("mail.smtp.password", password);

Authenticator authenticator = new Authenticator() {

@Override

protected PasswordAuthentication getPasswordAuthentication() {

// TODO Auto-generated method stub

return new PasswordAuthentication(username, password);

}

};

// 得到默认的对话对象

Session session = Session.getDefaultInstance(props, authenticator );

// 创建一个消息，并初始化该消息的各项元素

MimeMessage msg = new MimeMessage(session);

nick = MimeUtility.encodeText(nick);

msg.setFrom(new InternetAddress(nick + "<" + from + ">"));

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

String[] arrs = to.split(",");

if(arrs.length > 0){

InternetAddress[] addresses = new InternetAddress[arrs.length];

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

addresses[i] = new InternetAddress(arrs[i]);

}

msg.addRecipients(RecipientType.TO, addresses);

msg.setSubject(subject);

// 后面的BodyPart将加入到此处创建的Multipart中

Multipart mp = new MimeMultipart();

// 附件操作

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

MimeBodyPart tbp = new MimeBodyPart();

tbp.setText(body);

mp.addBodyPart(tbp);

for (String filepath : filepaths) {

MimeBodyPart mbp = new MimeBodyPart();

// 得到数据源

FileDataSource fds = new FileDataSource(filepath);

// 得到附件本身并至入BodyPart

mbp.setDataHandler(new DataHandler(fds));

// 得到文件名同样至入BodyPart

mbp.setFileName(fds.getName());

mp.addBodyPart(mbp);

}

filepaths.clear();

// Multipart加入到信件

msg.setContent(mp);

}else{

msg.setText(body);

}

// 设置信件头的发送日期

msg.setSentDate(new Date());

msg.saveChanges();

// 发送信件

Transport transport = session.getTransport("smtp");

transport.connect();

transport.sendMessage(msg, msg.getRecipients(RecipientType.TO));

transport.close();

return true;

}else{

logger.error("None recivers!");

return false;

}

}else{

logger.error("None recivers!");

return false;

}

}

}