中广有线枣庄分公司绩效考核系统

研究成果报告

**一、本项目研究的主要内容**

项目进行的主要内容分为数据库设计，后台逻辑代码编写，前台页面代码编写，数据导入脚本代码，模拟打分测试脚本代码编写。

1. 数据库设计

根据需求分析，数据库表结构设计如下：

部门评分记录表

department\_score\_record

department\_score\_record\_id 部门评分细则记录id

department\_from\_id 评分部门id from

department\_to\_id 给哪个部门打分

score\_item\_id 被评分的细则id

score 获得的分数

score\_date 打分日期

部门得分表

department\_score

department\_total\_score\_id 部门总得分id

department\_id 部门id

total\_score 部门总得分

annotation 修改分数注释

date 得分日期

部门评分细则表

department\_score\_item

score\_item\_id 评分细则id

score\_item\_name name

score\_description 评分细则描述

score\_value 评分细则分值

score\_reference基准值

score\_calculation\_method 计算方法

department\_id 所属部门id

score\_category\_id 所属类别id

score\_creation\_date 创建日期

score\_in\_use 目前可用状态

部门互评关联表

department\_score\_relation

department\_score\_relation\_id 部门互评关联id

score\_item\_id 关联评分细则id

department\_id 关联部门id

个人评分细则表

staff\_score\_item

score\_item\_id 评分细则id

score\_weights 评分细则权重

score\_description 评分细则描述

score\_standard 评分标准

score\_creation\_date 创建日期

score\_in\_use 目前可用状态

部门表 department

department\_id 部门id

departments\_name 部门名称

higher\_department\_id 上级部门ID

department\_description 部门描述

员工表 starff

staff\_id 员工号

staff\_name 员工姓名

staff\_sex 员工性别

staff\_password 登陆密码

department\_id 所属部门号

post\_id 职位

role\_id 所属角色号

部门条目类别表

score\_item\_category

score\_item\_category\_id 部门条目类别id

score\_item\_category\_name 部门条目名称

员工评分记录表

staff\_score\_record

staff\_score\_record\_id 个人评分记录id

scored\_from\_staff\_id 评分人id

scored\_to\_staff\_id 被评分员工号

score\_item\_id 个人评分细则id

score float, 得分

score\_date date

个人得分表

staff\_score

staff\_score\_id 个人得分id

staff\_id 被打分员工id

staff\_score 得分

annotation 修改分数注释

date 日期

角色表（控制页面） role

role\_id 角色号

role\_name 角色名称

role\_description 角色描述

页面分类 funcpage\_category

funcpage\_category\_id 页面分类号

funcpage\_category \_name 页面分类名称

页面 funcpage

funcpage\_id 页面号

funcpage\_name 页面名称

funcpage\_action action

funcpage\_category\_id 页面分类

职位表 post

post\_id 职位号

post\_name 职位名称

post\_description 职位描述

角色页面对照表

role\_funcpage\_relation

role\_funcpage\_relation\_id 角色页面关联id

role\_id 角色号

funcpage\_id 页面号

数据库建表sql代码：

-- 建表 外键未添加 auto\_increment也未添加

--

-- 删表

drop table if exists week\_meeting\_record;

drop table if exists month\_meeting\_record;

drop table if exists training\_plan;

drop table if exists upload\_file;

drop table if exists message;

drop table if exists salary;

drop table if exists department\_key\_work;

drop table if exists task\_progress;

drop table if exists configuration\_data;

drop table if exists staff\_score;

drop table if exists staff\_score\_record;

drop table if exists staff\_score\_relation;

drop table if exists staff\_score\_item;

drop table if exists department\_score;

drop table if exists department\_score\_record;

drop table if exists department\_score\_relation;

drop table if exists department\_score\_item;

drop table if exists score\_category;

drop table if exists staff;

drop table if exists staff\_class;

drop table if exists department;

drop table if exists post;

drop table if exists role\_funcpage\_relation;

drop table if exists funcpage;

drop table if exists funcpage\_category;

drop table if exists role;

drop table if exists upload\_file;

-- role表

create table role(

role\_id int(11) not null auto\_increment,

role\_name varchar(20) not null,

role\_description varchar(200),

primary key(role\_id)

) default charset=utf8;

-- 页面分类表

create table funcpage\_category(

funcpage\_category\_id int(11) not null auto\_increment,

funcpage\_category\_name varchar(200) not null,

primary key(funcpage\_category\_id)

)default charset=utf8;

-- 页面表　通过角色来配置

create table funcpage(

funcpage\_id int(11) not null auto\_increment,

funcpage\_name varchar(20) not null,

funcpage\_action text not null,

funcpage\_category\_id int(11),

primary key (funcpage\_id),

foreign key(funcpage\_category\_id) references funcpage\_category(funcpage\_category\_id)

)default charset=utf8;

-- role funcpage relationship

create table role\_funcpage\_relation(

role\_funcpage\_relation\_id int(11) not null auto\_increment,

role\_id int(11) not null,

funcpage\_id int(11) not null,

primary key (role\_funcpage\_relation\_id),

foreign key(role\_id) references role(role\_id),

foreign key(funcpage\_id) references funcpage(funcpage\_id)

)default charset=utf8;

-- post表

create table post(

post\_id int(11) not null auto\_increment,

post\_name varchar(20) not null,

primary key(post\_id)

) default charset=utf8;

-- department表

create table department(

department\_id int(11) not null auto\_increment,

department\_name varchar(20) not null,

higher\_department\_id int(11),

nature int(3) not null,

primary key(department\_id)

) default charset=utf8;

-- staff\_class

create table staff\_class(

class\_id char(9) not null,

class\_name varchar(200) not null,

class\_comment varchar(200),

primary key(class\_id)

)default charset=utf8;

-- staff表

create table staff (

staff\_id int(11) not null auto\_increment,

staff\_name varchar(20) not null,

staff\_no varchar(20) not null, -- gonghao

staff\_sex varchar(10),

staff\_password varchar(20) not null,

department\_id int(11) not null,

post\_id int(11) not null,

role\_id int(11) not null,

class\_id text, -- string ;;

primary key (staff\_id),

foreign key(role\_id) references role(role\_id),

foreign key(post\_id) references post(post\_id),

foreign key(department\_id) references department(department\_id)

) default charset=utf8;

-- 条目类别表

create table score\_category(

score\_category\_id int(11) not null auto\_increment,

score\_category\_name varchar(20),

primary key(score\_category\_id)

)default charset=utf8;

-- 部门条目表

create table department\_score\_item(

score\_item\_id int(11) not null auto\_increment,

score\_item\_name varchar(200),

score\_description varchar(200),

score\_value float,

score\_reference varchar(200),

score\_calculation\_method text,

department\_id int(11),

score\_category\_id int(11),

score\_creation\_date date,

score\_in\_use int(1),

primary key(score\_item\_id),

foreign key(department\_id) references department(department\_id),

foreign key(score\_category\_id) references score\_category(score\_category\_id)

)default charset=utf8;

-- 部门条目对照表

create table department\_score\_relation(

department\_score\_relation\_id int(11) not null auto\_increment,

score\_item\_id int(11),

department\_id int(11),

primary key(department\_score\_relation\_id),

foreign key(score\_item\_id) references department\_score\_item(score\_item\_id),

foreign key(department\_id) references department(department\_id)

)default charset=utf8;

-- 部门打分记录表

create table department\_score\_record(

department\_score\_record\_id int(11) not null auto\_increment,

department\_from\_id int(11),

department\_to\_id int(11),

score\_item\_id int(11),

score float,

score\_comment text,

score\_date date,

which\_year int(11),

which\_month int(11),

modify\_times int(11),

owner int(11), -- staffId

primary key(department\_score\_record\_id),

foreign key(department\_from\_id) references department(department\_id),

foreign key(department\_to\_id) references department(department\_id),

foreign key(score\_item\_id) references department\_score\_item(score\_item\_id),

foreign key(owner) references staff(staff\_id)

)default charset=utf8;

-- 部门得分表

create table department\_score(

department\_total\_score\_id int(11) not null auto\_increment,

department\_id int(11),

total\_score float,

score\_date date,

which\_year int(11),

which\_month int(11),

annotation text,

owner int(11), -- staffId

primary key(department\_total\_score\_id),

foreign key(department\_id) references department(department\_id),

foreign key(owner) references staff(staff\_id)

)default charset=utf8;

-- 个人评分条目表

create table staff\_score\_item(

score\_item\_id int(11) not null auto\_increment,

score\_item\_name varchar(20),

score\_weight float,

score\_description varchar(200),

score\_standard1 varchar(200),

score\_standard2 varchar(200),

score\_standard3 varchar(200),

score\_standard4 varchar(200),

score\_standard5 varchar(200),

score\_category\_id int(11) not null,

score\_creation\_date date,

score\_in\_use int(1),

primary key(score\_item\_id),

foreign key(score\_category\_id) references score\_category(score\_category\_id)

)default charset=utf8;

-- 个人条目对照表

create table staff\_score\_relation(

staff\_score\_relation\_id int(11) not null auto\_increment,

score\_item\_id int(11) not null ,

department\_id int(11) not null ,

post\_id int(11) not null ,

primary key(staff\_score\_relation\_id),

foreign key(score\_item\_id) references staff\_score\_item(score\_item\_id),

foreign key(department\_id) references department(department\_id),

foreign key(post\_id) references post(post\_id)

)default charset=utf8;

-- 员工打分记录表

create table staff\_score\_record(

staff\_score\_record\_id int(11) not null auto\_increment,

scored\_from\_staff\_id int(11),

scored\_to\_staff\_id int(11),

score\_item\_id int(11),

score float,

score\_date date,

which\_year int(11),

which\_month int(11),

modify\_times int(11),

owner int(11), -- staffId

primary key(staff\_score\_record\_id),

foreign key(scored\_from\_staff\_id) references staff(staff\_id),

foreign key(scored\_to\_staff\_id) references staff(staff\_id),

foreign key(score\_item\_id) references staff\_score\_item(score\_item\_id),

foreign key(owner) references staff(staff\_id)

)default charset=utf8;

-- 个人得分表

create table staff\_score(

staff\_score\_id int(11) not null auto\_increment,

staff\_id int(11),

staff\_score float,

score\_date date,

which\_year int(11),

which\_month int(11),

annotation text,

owner int(11), -- staffId

primary key(staff\_score\_id),

foreign key(staff\_id) references staff(staff\_id),

foreign key(owner) references staff(staff\_id)

)default charset=utf8;

-- 参数表

create table configuration\_data(

configuration\_data\_id int not null auto\_increment,

configuration\_data\_key varchar(200),

configuration\_data\_value varchar(200),

primary key(configuration\_data\_id)

)default charset=utf8;

-- 任务进度表

create table task\_progress(

task\_progress\_id int not null auto\_increment,

task\_progress float,

creation\_date date,

which\_year int(11),

which\_month int(11),

primary key(task\_progress\_id)

)default charset=utf8;

-- 部门阶段重点工作表

create table department\_key\_work(

key\_work\_id int not null auto\_increment,

department\_id int not null,

key\_work\_content text not null,

start\_year int not null,

start\_month int not null,

end\_year int not null,

end\_month int not null,

primary key(key\_work\_id),

foreign key(department\_id) references department(department\_id)

)default charset=utf8;

-- 工资表

create table salary(

salary\_id int not null auto\_increment,

staff\_id int not null,

salary float not null,

primary key(salary\_id)

)default charset=utf8;

-- 消息表

create table message(

message\_id int not null auto\_increment,

title varchar(100) not null,

content longtext not null,

simple\_content text,

message\_type int,

sender int,

receiver text,

received text,

attachment text,

message\_date DATETIME,

primary key(message\_id),

foreign key(sender) references staff(staff\_id)

)default charset=utf8;

-- 文件关联

create table upload\_file(

file\_id varchar(50),

file\_name varchar(100) not null,

file\_type varchar(100) not null,

file\_size int not null,

creator\_id int(11) not null,

upload\_time DATETIME,

primary key(file\_id)

)default charset=utf8;

-- 周会议记录

create table week\_meeting\_record(

week\_meeting\_record\_id int not null auto\_increment,

week\_meeting\_record\_content longtext,

upload\_time DATETIME,

upload\_staff int(11),

upload\_department int(11),

primary key(week\_meeting\_record\_id),

foreign key(upload\_staff) references staff(staff\_id),

foreign key(upload\_department) references department(department\_id)

)default charset=utf8;

-- 月会议记录

create table month\_meeting\_record(

month\_meeting\_record\_id int not null auto\_increment,

month\_meeting\_record\_content longtext,

upload\_time DATETIME,

upload\_staff int(11),

upload\_department int(11),

primary key(month\_meeting\_record\_id),

foreign key(upload\_staff) references staff(staff\_id),

foreign key(upload\_department) references department(department\_id)

)default charset=utf8;

-- 培训计划

create table training\_plan(

training\_plan\_id int not null auto\_increment,

training\_plan\_content longtext,

upload\_time DATETIME,

upload\_staff int(11),

upload\_department int(11),

primary key(training\_plan\_id),

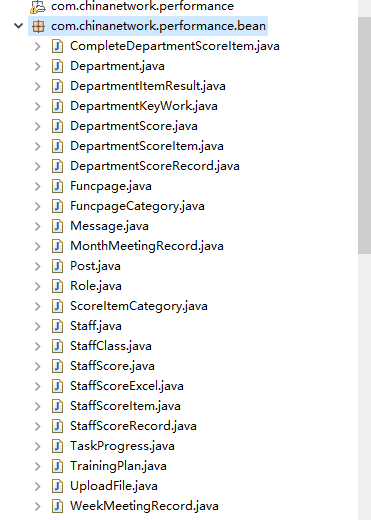
foreign key(upload\_staff) references staff(staff\_id),

foreign key(upload\_department) references department(department\_id)

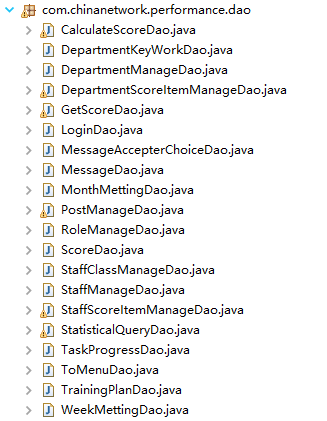
)default charset=utf8;

1. 后台逻辑代码编写

后台代码文件结构：

Bean文件为数据库表映射对象：

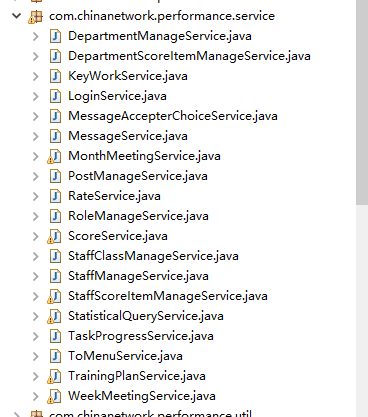
Dao文件为数据库操作对象：



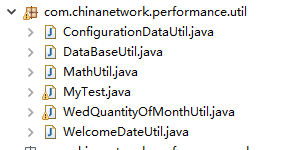
Filter文件为基本过滤器对象：



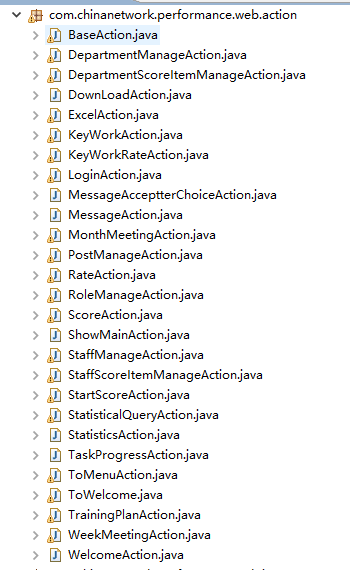
Service文件为后台逻辑对象：



Utils文件为通用工具对象：



Action文件为后台于前台交互映射action对象：

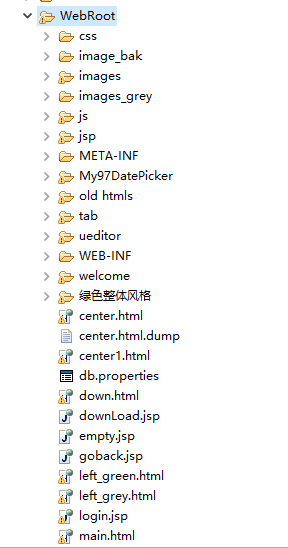


Struts文件为struts2框架需要的前台访问请求向后台action对象映射的文件：



1. 前台页面代码编写

前台页面采用jsp编写，具体文件结构如下



其中使用的第三方库My97DatePicker和ueditor，分别用来实现系统中的时间选取和文本输入。

1. 数据导入脚本代码编写

初期基本数据的导入采用excel表通过脚本直接导入数据库的方式。方便批量修改和数据的创建。导入excel文件的脚本代码如下：

# -\*- coding: utf-8 -\*-

import MySQLdb

import xlrd

def insert(tablename, colnames, values):

print 'inserting ',tablename

try:

coon = MySQLdb.connect(host='10.7.0.124', user='root', passwd='1234', db='pfe', port=3306, charset='utf8')

cur = coon.cursor()

count = cur.execute('desc %s'%tablename)

results = cur.fetchall()

colnamestr = results[0][0]

coltypestr = '%s'

for i in range(1, count):

colname = results[i][0]

coltype = results[i][1]

colnamestr = colnamestr+', '+colname

coltypestr = coltypestr+', %s'

sql = 'insert into '+tablename+'('+colnamestr+') '+'values('+coltypestr+')'

cur.executemany(sql, values)

coon.commit()

coon.close()

except MySQLdb.Error, e:

print 'mysql error %d, %s'%(e.args[0], e.args[1])

def func(filename):

book = xlrd.open\_workbook(filename)

sheets = book.sheets()

for sheet in sheets:

tablename = sheet.name

colnames =[]

for cell in sheet.row(0):

if cell.ctype != 0:

colnames.append(cell.value)

colnum = len(colnames)

values = []

for n in range(1, sheet.nrows):

realrow = []

for m in range(0, colnum):

realrow.append(sheet.cell(n, m).value)

values.append(realrow)

insert(tablename, colnames, values )

if \_\_name\_\_ == '\_\_main\_\_':

func ('excel.xls')

func ('excel2.xls')

其中使用了第三方库xlrd和MySQLdb分别用来读取excel文件和写入数据库。

1. 模拟打分测试脚本代码编写

系统代码编写基本完成以后需要系统得对系统进行测试。但是由于公司部门人员众多，人工打分会非常费时费力。由此，使用sql存储过程编写了模拟打分功能。方便逻辑代码的测试。

**二、关键技术及创新点**

本系统在编写过程中使用的关键技术如下：

1. 数据库操作代码DataBaseUtil.java

package com.chinanetwork.performance.util;

import java.sql.Connection;

import java.sql.DriverManager;

import java.sql.PreparedStatement;

import java.sql.ResultSet;

import java.sql.SQLException;

import java.util.List;

public class DataBaseUtil {

private static String driver = "com.mysql.jdbc.Driver";

private static String url = "jdbc:mysql://127.0.0.1:3306/pfe?useUnicode=true&characterEncoding=utf8";

private static String username = "root";

private static String password = "liugang";

public static Connection getConnection() throws Exception {

Connection connection = null;

try {

Class.forName(driver);

connection = DriverManager.getConnection(url, username, password);

} catch (SQLException e) {

e.printStackTrace();

throw new RuntimeException("getConnection error");

}

return connection;

}

public static void close(ResultSet resultSet,

PreparedStatement preparedStatement, Connection connection)

throws Exception {

try {

if (resultSet != null) {

resultSet.close();

}

if (preparedStatement != null) {

preparedStatement.close();

}

if (connection != null) {

connection.close();

}

} catch (SQLException e) {

e.printStackTrace();

throw new RuntimeException("close error");

}

}

public static void close(PreparedStatement preparedStatement,

Connection connection) throws Exception {

try {

if (preparedStatement != null) {

preparedStatement.close();

}

if (connection != null) {

connection.close();

}

} catch (SQLException e) {

e.printStackTrace();

throw new RuntimeException("close error");

}

}

public static void main(String[] args) throws Exception {

System.out.println(DataBaseUtil.getConnection());

}

public static void executeSQL(String sql) throws Exception {

Connection conn = null;

PreparedStatement ps = null;

conn = DataBaseUtil.getConnection();

ps = conn.prepareStatement(sql);

ps.executeUpdate();

DataBaseUtil.close(ps, conn);

}

public static String integerListToSQLString(List<Integer> list) {

String sql = "";

if (list != null) {

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

sql.concat(String.valueOf(list.get(i)) + ";");

}

}

return sql;

}

}

1. 算分核心算法代码ScoreService.java

package com.chinanetwork.performance.service;

/\*\*

\* 负责计算得分的业务逻辑层

\*/

import java.sql.Date;

import java.text.DecimalFormat;

import java.text.ParseException;

import java.text.SimpleDateFormat;

import java.util.ArrayList;

import java.util.Calendar;

import java.util.HashMap;

import java.util.Iterator;

import java.util.List;

import java.util.Map;

import com.chinanetwork.performance.bean.CompleteDepartmentScoreItem;

import com.chinanetwork.performance.bean.Department;

import com.chinanetwork.performance.bean.DepartmentKeyWork;

import com.chinanetwork.performance.bean.DepartmentScore;

import com.chinanetwork.performance.bean.DepartmentScoreItem;

import com.chinanetwork.performance.bean.DepartmentScoreRecord;

import com.chinanetwork.performance.bean.Staff;

import com.chinanetwork.performance.bean.StaffScore;

import com.chinanetwork.performance.bean.StaffScoreExcel;

import com.chinanetwork.performance.bean.StaffScoreRecord;

import com.chinanetwork.performance.dao.CalculateScoreDao;

import com.chinanetwork.performance.dao.GetScoreDao;

import com.chinanetwork.performance.dao.PostManageDao;

import com.chinanetwork.performance.dao.ScoreDao;

import com.chinanetwork.performance.dao.StaffManageDao;

import com.chinanetwork.performance.dao.TaskProgressDao;

import com.chinanetwork.performance.util.ConfigurationDataUtil;

import com.chinanetwork.performance.util.MathUtil;

/\*

\* 算分

\*

\*/

public class ScoreService {

private GetScoreDao getScoreDao=new GetScoreDao();

private PostManageDao postManageDao = new PostManageDao();

private ScoreDao scoreDao = new ScoreDao();

//=============================================================以下向部门重点工作打分提供服务================================================================

//==========================添加功能，在重点工作打分页面，显示重点工作列表===========================

/\*\*

\* 重点工作条目打分

\* @param staff

\* @param itemsIds

\* @param departmentScores

\* @return

\*/

public boolean submitDepartmentWorkScore(Staff staff,List<String> itemsIds,List<String> departmentScores){

try{

Department departmentFrom=getScoreDao.getDepartmentFromStaff(staff);

List<Department> departmentTos=new ArrayList<Department>();

for(String s:itemsIds){

int itemsId=Integer.parseInt(s);

departmentTos.add(getScoreDao.getDepartmentFromItemId(itemsId));

}

Date date=new Date(System.currentTimeMillis());

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

int itemsId=Integer.parseInt(itemsIds.get(i));

float departmentScore=Float.parseFloat(departmentScores.get(i));

getScoreDao.submitDepartmentWorkScore(departmentFrom.getDepartmentId(), departmentTos.get(i).getDepartmentId(), itemsId, departmentScore, date);

}

return true;

}catch(Exception e){

e.printStackTrace();

}

return false;

}

//填写修改次数

public void setModifyKeyWorkScoreTimes(String times,int fDepId,int tDepId,String category)throws Exception{

int time=Integer.parseInt(times);

try{

getScoreDao.insertModifyKeyWorkScoreTimes(time, fDepId, tDepId, category);

}catch(Exception e){

e.printStackTrace();

}

}

//=============================================================以下向重点得分修改提供服务=====================================================================

/\*\*

\* 当用户进入“重点工作打分修改界面”时，需要“已打分部门”下拉列表

\* 该方法提供这样的功能

\*/

public List<Department> getBeScoredDepartmentList()throws Exception{

//获取绩效考评的开始和结束时间

int year=ConfigurationDataUtil.getIntegerVlaue("which\_year");

int month=ConfigurationDataUtil.getIntegerVlaue("which\_month");

//获取“重点工作”被打分完毕的部门列表

List<Department> departmentsScored=new ArrayList<Department>();

departmentsScored=getScoreDao.getkeyWorkNowBeScoredDepartmentsModify(year, month);

return departmentsScored;

}

//-----------------------------------------------------------------------------------------------------------------------------------

/\*\*

\* 根据在下拉列表中选择的“部门名称”，返回其在打分就表中的对应的记录，包括分数

\* 改进方法

\*/

public List<CompleteDepartmentScoreItem>getDepartmentWorkBeScoredItem(String departmentIdStr)throws Exception{

int departmentId=Integer.parseInt(departmentIdStr);

//获取绩效考评的开始和结束时间

int year=ConfigurationDataUtil.getIntegerVlaue("which\_year");

int month=ConfigurationDataUtil.getIntegerVlaue("which\_month");

List<CompleteDepartmentScoreItem> completeDepartmentScoreItems=getScoreDao.getDepartmentWorkBeScoredItem(departmentId, year, month);

return completeDepartmentScoreItems;

}

//-----------------------------------------------------------------------------------------------------------------------------------

/\*\*

\* 为修改部门重点工作得分提供服务

\* @param itemsIds

\* @param departmentScores

\* @return

\* @throws Exception

\*/

public boolean modifyDepartmentWorkScoreRecord(List<String>itemsIds ,List<String> departmentScores)throws Exception{

boolean isSuccess=false;

//获取绩效考评的开始和结束时间

long startTime=ConfigurationDataUtil.getLongVlaue("start\_time");

long endTime=ConfigurationDataUtil.getLongVlaue("end\_time");

//修改相应评分记录中的分数

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

isSuccess=getScoreDao.modifyDepartmentWorkScoreRecord(Integer.parseInt(itemsIds.get(i)), startTime, endTime, Float.parseFloat(departmentScores.get(i)));

}

return isSuccess;

}

//------------------------------------------------------------------------------------

/\*\*

\* 为修改部门重点工作得分提供服务

\* @param itemsIds

\* @param departmentScores

\* @return

\* @throws Exception

\*/

public boolean modifyDepartmentWorkScoreRecord(Staff staff,List<String>itemsIds ,List<String> departmentScores)throws Exception{

boolean isSuccess=false;

//获取绩效考评的开始和结束时间

int year=ConfigurationDataUtil.getIntegerVlaue("which\_year");

int month=ConfigurationDataUtil.getIntegerVlaue("which\_month");

//修改相应评分记录中的分数

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

isSuccess=getScoreDao.modifyDepartmentWorkScoreRecord(staff,Integer.parseInt(itemsIds.get(i)), year, month, Float.parseFloat(departmentScores.get(i)));

}

return isSuccess;

}

//------------------------------------------------------------------------------------

//==================================================================以下给部门互评得分提供服务======================================================================

/\*\*

\* 判断部门互评的评分过程是否结束

\* @return

\* @throws Exception

\*/

public List<String> confirmAccomplishDepartmentScore()throws Exception{

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

//获取绩效考评的开始和结束时间

long startTime=ConfigurationDataUtil.getLongVlaue("start\_time");

long endTime=ConfigurationDataUtil.getLongVlaue("end\_time");

Map<Department,List<DepartmentScoreItem>> notScoredMap=new HashMap<Department,List<DepartmentScoreItem>>();

notScoredMap=postManageDao.getNotScoredDepartmentScoreItem(startTime, endTime);

System.out.println(notScoredMap.size());

if(notScoredMap.size()>0){

Iterator iterator=notScoredMap.keySet().iterator();

while(iterator.hasNext()){

Department departmentKey=(Department)iterator.next();

List<DepartmentScoreItem> departmentScoreItems=notScoredMap.get(departmentKey);

if(departmentScoreItems==null){

String message=departmentKey.getDepartmentName()+"还没有开始评分";

messageList.add(message);

}else{

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

String message=departmentKey.getDepartmentName()+"对"+departmentScoreItems.get(i).getDepartment().getDepartmentName()+"的评分还没有结束";

messageList.add(message);

}

}

}

}

return messageList;

}

//====================================================================以下给员工得分提供服务=================================================================

/\*\*

\* 确定员工打分过程是否结束，如果没有结束，提供具体的细节信息

\*/

public List<String> confirmAccomplishStaffScore() throws Exception{

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

//获取绩效考评的开始和结束时间

int year=ConfigurationDataUtil.getIntegerVlaue("which\_year");

int month=ConfigurationDataUtil.getIntegerVlaue("which\_month");

//获取拥有“员工打分”权限的员工，除管理员之外

List<Staff> ownScorePermissionsStaffList=postManageDao.getOwnScorePermissionsStaffs();

//循环员工列表

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

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

Staff ownScorePermissionsStaff=ownScorePermissionsStaffList.get(i);

//该员工打分对象列表

List<Staff> beScoredStaffList=null;

beScoredStaffList = postManageDao.getScoreStaffsByPostId(ownScorePermissionsStaff);

if(beScoredStaffList == null )

{

continue;

}

//循环打分对象列表

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

Staff beScoreStaff=beScoredStaffList.get(j);

//获得该打分对象的评分细则的打分记录的总条数

int beScoredQuantity=getScoreDao.getStaffScoreRecordQuantity(ownScorePermissionsStaff,beScoreStaff, year,month);

//获得该打分对象的评分细则的总条数

int scoreItemQuantity=getScoreDao.getStaffScoreItemQuantity(beScoreStaff);

System.out.println(ownScorePermissionsStaff.getStaffName());

System.out.println("打分记录："+beScoredQuantity);

System.out.println("应打分记录："+scoreItemQuantity);

//判断该打分对象的评分过程是否结束

if(beScoredQuantity==scoreItemQuantity){

//结束

continue;

}else{

//没有结束

String message=ownScorePermissionsStaff.getDepartment().getDepartmentName()+ownScorePermissionsStaff.getStaffName()+"对"+beScoreStaff.getStaffName()+"的评分还没有结束";

messageList.add(message);

}

}

}

return messageList;

}

//----------------------------(2)部门计算分数，应该只能计算一次

/\*\*

\* 确定“部门算分”是否已经执行过

\*/

public boolean confirmOnlyCalculateDepartmentScore()throws Exception{

//获取绩效年份和月份

int year=ConfigurationDataUtil.getIntegerVlaue("which\_year");

int month=ConfigurationDataUtil.getIntegerVlaue("which\_month");

int quantity=getScoreDao.getNowDepartmentScoreQuantity(year, month);

if(quantity>=1){

return false;

}else{

return true;

}

}

//---------------------------(3)员工计算分数，应该只能计算一次

/\*\*

\* 确定“员工算分”是否已经执行过

\*/

public boolean confirmOnlyCalculateStaffScore()throws Exception{

//获取绩效年份和月份

int year=ConfigurationDataUtil.getIntegerVlaue("which\_year");

int month=ConfigurationDataUtil.getIntegerVlaue("which\_month");

int quantity=getScoreDao.getNowStaffScoreQuantity(year, month);

if(quantity>=1){

return false;

}else{

return true;

}

}

//----------------------------(4)员工算分之前，应该录入“任务进度”

/\*\*

\* 确定“任务进度”是否已经被录入

\*/

public boolean confirmTaskProgressInput()throws Exception{

int year = ConfigurationDataUtil.getIntegerVlaue("which\_year");

int month = ConfigurationDataUtil.getIntegerVlaue("which\_month");

boolean flag=false;

TaskProgressService taskProgressService = new TaskProgressService();

flag=taskProgressService.check(year, month);

return flag;

}

public CalculateScoreDao calculateScoreDao = new CalculateScoreDao();

public WeekMeetingService weekMeetingService = new WeekMeetingService();

// ----------------------------------------------------------------------------------------------------------------------部门绩效计算开始

public boolean calculateDepartmentScore() throws Exception {

try {

// 计算市公司各部门，各区营业部的部门的得分

calculateFirstDepartmentScore();

return true;

} catch (Exception e) {

return false;

}

}

/\*\*

\* First：市公司各部门，各区营业部的部门得分

\*

\* @throws Exception

\*/

@SuppressWarnings("unchecked")

private void calculateFirstDepartmentScore() throws Exception {

try {

// 修改

int year = ConfigurationDataUtil.getIntegerVlaue("which\_year");

int month = ConfigurationDataUtil.getIntegerVlaue("which\_month");

// map<department\_to\_id, score>

Map<Integer, Float> scoreList = calculateScoreDao

.fuckDepartmentScore(year, month);

// 周例会检测，加分

scoreList = weekMeetingService

.updateDepartmentScoreByWeekAndMonthMeeting(scoreList,

year, month);

getScoreDao.insertDepartmentScore(scoreList, year, month);

//calculateScoreDao.fuckComment(year, month);

System.out.println("caonima");

System.out.println("caonima");

System.out.println("caonima");

System.out.println("caonima");

System.out.println("caonima");

System.out.println("caonima");

} catch (Exception e) {

e.printStackTrace();

}

}

/\*\*

\* 市公司各部门副主任和员工，以及各区营业部的各个科室副科长和科员的绩效考评的计算公式

\*

\* @param staff

\* 被打分的员工

\* @param staffScoreMap

\* 员工各项评分细则的得分Map，以打分者的职位不同分类

\* @param year

\* 绩效考评年份

\* @param month

\* 绩效考评月份

\* @param fStaffA

\* 公式参数，A打分员工的职位名称

\* @param fStaffB

\* 公式参数，B打分员工的职位名称

\* @param coefficientA

\* 公式参数，A打分员工对应的权重

\* @param coefficientB

\* 公式参数，B打分员工对应的权重

\* @param staffScoreCoefficient

\* 公式参数，员工评分得分所占的权重

\* @param depScoreCoefficient

\* 公式参数，员工所属的部门得分所占的权重

\* @return totalScore 返回根据公式计算出的分数

\* @throws Exception

\*/

public float calculateStaff(Staff staff, Map<Staff, Float> staffScoreMap,

int year, int month, String fStaffA, String fStaffB,

float coefficientA, float coefficientB,

float staffScoreCoefficient, float depScoreCoefficient)

throws Exception {

float totalScore = 0;

Iterator iterator = staffScoreMap.keySet().iterator();

List<Float> scoreListA = new ArrayList<Float>();

List<Float> scoreListB = new ArrayList<Float>();

while (iterator.hasNext()) {

Staff staffKey = (Staff) iterator.next();

System.out.println("打分员工名称：" + staffKey.getStaffName());

System.out.println("所打分数：" + staffScoreMap.get(staffKey));

if (staffKey.getPost().getPostName().equals(fStaffA)) {

scoreListA.add(staffScoreMap.get(staffKey));

// totalScore=totalScore+(float)(staffScoreMap.get(staffKey)\*coefficientA);

} else if (staffKey.getPost().getPostName().equals(fStaffB)) {

// totalScore=totalScore+(float)(staffScoreMap.get(staffKey)\*coefficientB);

scoreListB.add(staffScoreMap.get(staffKey));

}

}

float totalA = 0;

float totalB = 0;

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

totalA += scoreListA.get(i);

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

totalB += scoreListB.get(i);

totalA = totalA / scoreListA.size();

totalB = totalB / scoreListB.size();

totalScore = coefficientA \* totalA + coefficientB \* totalB;

System.out.println(fStaffA + "打分为：" + totalA);

System.out.println(fStaffB + "打分为：" + totalB);

System.out.println("员工 " + staff.getStaffName() + " 的个人得分为: "

+ coefficientA + "\*" + totalA + " + " + coefficientB + "\*"

+ totalB + " = " + totalScore);

DepartmentScore departmentScore = calculateScoreDao

.getScoreOfDepartmentBelongStaff(staff.getDepartment()

.getDepartmentId(), year, month);

System.out.println("所属部门："

+ departmentScore.getDepartment().getDepartmentName());

System.out.println("所属部门分数：" + departmentScore.getScore());

float fuck = totalScore;

totalScore = totalScore \* staffScoreCoefficient

+ departmentScore.getScore() \* depScoreCoefficient;

System.out.println("员工 " + staff.getStaffName() + " 部门加权得分为: " + fuck

+ "\*" + staffScoreCoefficient + " + "

+ departmentScore.getScore() + "\*" + depScoreCoefficient

+ " = " + totalScore);

return totalScore;

}

// --------------------------------------------------------------------------------------------------------------------------------部门绩效计算结束

public List<Float> calculateStaff2(Staff staff,

Map<Staff, Float> staffScoreMap, int year, int month,

String fStaffA, String fStaffB, float coefficientA,

float coefficientB, float staffScoreCoefficient,

float depScoreCoefficient) throws Exception {

float totalScore = 0;

Iterator iterator = staffScoreMap.keySet().iterator();

List<Float> scoreListA = new ArrayList<Float>();

List<Float> scoreListB = new ArrayList<Float>();

while (iterator.hasNext()) {

Staff staffKey = (Staff) iterator.next();

System.out.println("打分员工名称：" + staffKey.getStaffName());

System.out.println("所打分数：" + staffScoreMap.get(staffKey));

if (staffKey.getPost().getPostName().equals(fStaffA)) {

scoreListA.add(staffScoreMap.get(staffKey));

// totalScore=totalScore+(float)(staffScoreMap.get(staffKey)\*coefficientA);

} else if (staffKey.getPost().getPostName().equals(fStaffB)) {

// totalScore=totalScore+(float)(staffScoreMap.get(staffKey)\*coefficientB);

scoreListB.add(staffScoreMap.get(staffKey));

}

}

float totalA = 0;

float totalB = 0;

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

totalA += scoreListA.get(i);

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

totalB += scoreListB.get(i);

totalA = totalA / scoreListA.size();

totalB = totalB / scoreListB.size();

totalScore = coefficientA \* totalA + coefficientB \* totalB;

System.out.println(fStaffA + "打分为：" + totalA);

System.out.println(fStaffB + "打分为：" + totalB);

System.out.println("员工 " + staff.getStaffName() + " 的个人得分为: "

+ coefficientA + "\*" + totalA + " + " + coefficientB + "\*"

+ totalB + " = " + totalScore);

DepartmentScore departmentScore = calculateScoreDao

.getScoreOfDepartmentBelongStaff(staff.getDepartment()

.getDepartmentId(), year, month);

System.out.println("所属部门："

+ departmentScore.getDepartment().getDepartmentName());

System.out.println("所属部门分数：" + departmentScore.getScore());

float fuck = totalScore;

totalScore = totalScore \* staffScoreCoefficient

+ departmentScore.getScore() \* depScoreCoefficient;

System.out.println("员工 " + staff.getStaffName() + " 部门加权得分为: " + fuck

+ "\*" + staffScoreCoefficient + " + "

+ departmentScore.getScore() + "\*" + depScoreCoefficient

+ " = " + totalScore);

List<Float> ret = new ArrayList<Float>(3);

ret.add(0, MathUtil.baoliu(fuck, 2));

ret.add(1, MathUtil.baoliu(totalScore, 2));

return ret;

}

public boolean calculateStaffScore() {

try {

List<Staff> staffList = null;

StaffManageDao staffManageDao = new StaffManageDao();

staffList = staffManageDao.getAllStaff();

/\*

\* //找到市场开发部 Department

\* departmentMarket=staffManageDao.getMarketDepartment();

\* //从员工列表中去掉市场开发部的员工（主任，副主任，员工） for(int

\* i=staffList.size()-1;i>=0;i--){ Staff staff=staffList.get(i);

\* if(staff

\* .getDepartment().getDepartmentId()==departmentMarket.getDepartmentId

\* ()){ staffList.remove(i); } }

\*/

List<StaffScore> staffScoreList = new ArrayList<StaffScore>();

List<Staff> postAStaffList = new ArrayList<Staff>(); // 副总经理List

List<Staff> postBStaffList = new ArrayList<Staff>(); // 市公司部门主任List

List<Staff> postCStaffList = new ArrayList<Staff>(); // 市公司部门副主任,市公司员工,区营业部科室副科长,区营业部科员工List

// List<Staff> postDStaffList=new ArrayList<Staff>(); //区营业部经理List

// List<Staff> postEStaffList=new ArrayList<Staff>(); //区营业部副经理List

// List<Staff> postFStaffList=new ArrayList<Staff>(); //区营业部科室科长List

// List<Staff> postGStaffList=new ArrayList<Staff>(); //总经理List

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

Staff staff = staffList.get(i);

String postName = staff.getPost().getPostName();

if (postName.equals("副总经理")) {

postAStaffList.add(staff);

} else if (postName.equals("市公司部门主任")) {

postBStaffList.add(staff);

} else if (postName.equals("市公司部门副主任")

|| postName.equals("市公司员工")) {

postCStaffList.add(staff);

}

staffList.remove(staff);

i--;

}

List<StaffScore> postAStaffScoreList = new ArrayList<StaffScore>(); // 副总经理List

List<StaffScore> postBStaffScoreList = new ArrayList<StaffScore>(); // 市公司部门主任List

List<StaffScore> postCStaffScoreList = new ArrayList<StaffScore>(); // 市公司部门副主任,市公司员工,区营业部科室副科长,区营业部科员工List

// List<StaffScore> postDStaffScoreList=new ArrayList<StaffScore>();

// //区营业部经理List

// List<StaffScore> postEStaffScoreList=new ArrayList<StaffScore>();

// //区营业部副经理List

// List<StaffScore> postFStaffScoreList=new ArrayList<StaffScore>();

// //区营业部科室科长List

// List<StaffScore> postGStaffScoreList=new ArrayList<StaffScore>();

// //总经理List

postBStaffScoreList = calculateTheScoreOfChief(postBStaffList);

postAStaffScoreList = calculateTheScoreOfDeputyManager(

postAStaffList, postBStaffScoreList);

postCStaffScoreList = calculateTheScoreOfDeputyDirectorOrDepEmpOrDeputyChiefOrClerk(postCStaffList);

// postDStaffScoreList=calculateTheScoreOfSalesManager(postDStaffList);

// postEStaffScoreList=calculateTheScoreOfSalesDeputyManager(postEStaffList);

// postFStaffScoreList=calculateTheScoreOfDirector(postFStaffList);

// /postGStaffScoreList=calculateLastStaffScore(postGStaffList,postAStaffScoreList);

staffScoreList.addAll(postAStaffScoreList);

staffScoreList.addAll(postBStaffScoreList);

staffScoreList.addAll(postCStaffScoreList);

// staffScoreList.addAll(postDStaffScoreList);

// staffScoreList.addAll(postEStaffScoreList);

// staffScoreList.addAll(postFStaffScoreList);

// staffScoreList.addAll(postGStaffScoreList);

calculateScoreDao.insertDataIntoStaffScore(staffScoreList);

return true;

} catch (Exception e) {

e.printStackTrace();

return false;

}

}

// --------------------------------------------------------------------------------------------------------------------------------部室主任算分开始

/\*\*

\* 计算市公司各部门主任的绩效考评得分

\*

\* @param staffList

\* 市公司各部门主任列表

\* @return staffScoreList 市公司各部门主任的得分列表

\* @throws Exception

\*/

private List<StaffScore> calculateTheScoreOfChief(List<Staff> staffList)

throws Exception {

System.out

.println("-------------------------------------------------------------------------------------------------");

System.out.println("部室主任计算分数开始");

int year = ConfigurationDataUtil.getIntegerVlaue("which\_year");

int month = ConfigurationDataUtil.getIntegerVlaue("which\_month");

List<StaffScore> staffScoreList = new ArrayList<StaffScore>();

float task = new TaskProgressDao().getTaskProgress(year, month)

.getTaskProgress();

System.out.println("任务进度：" + task);

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

Staff staff = staffList.get(i);

System.out.println("主任名称：" + staff.getStaffName());

float totalScore = 0;

// float task=calculateScoreDao.selectTaskSchedule(year, month);

// task=task\*100;

String explain = "";// 打分说明

DepartmentScore departmentScore = calculateScoreDao

.getScoreOfDepartmentBelongStaff(staff.getDepartment()

.getDepartmentId(), year, month);

System.out.println("部室名称："

+ staff.getDepartment().getDepartmentName());

System.out.println("部室得分：" + departmentScore.getScore());

if (departmentScore.getInstruction() != null

&& (!departmentScore.getInstruction().equals(""))) {

explain = explain + departmentScore.getInstruction();

}

System.out.println("部室性质："

+ departmentScore.getDepartment().getNature());

if (departmentScore.getDepartment().getNature() == 0) {

totalScore = task \* 0.7f + departmentScore.getScore() \* 0.3f;

} else if (departmentScore.getDepartment().getNature() == 1) {

totalScore = task \* 0.5f + departmentScore.getScore() \* 0.5f;

}

System.out.println("最后得分：" + totalScore);

DecimalFormat df = new DecimalFormat("#.00");

totalScore = Float.parseFloat(df.format(totalScore));

StaffScore staffScore = new StaffScore();

staffScore.setStaff(staff);

staffScore.setTotalScore(totalScore);

staffScore.setYear(year);

staffScore.setMonth(month);

staffScore.setInstruction(explain);

staffScoreList.add(staffScore);

}

System.out.println("部室主任计算分数结束");

return staffScoreList;

}

// --------------------------------------------------------------------------------------------------------------------------------员工算分结束

// --------------------------------------------------------------------------------------------------------------------------------员工算分开始

/\*\*

\* 计算市公司各部门副主任和员工，以及各区营业部的各个科室副科长和科员的绩效考评分数

\*

\* @param staffList

\* 市公司各部门副主任和员工，以及各区营业部的各个科室副科长和科员的集合列表

\* @return staffScoreList 员工绩效考评分数集合列表

\* @throws Exception

\*/

private List<StaffScore> calculateTheScoreOfDeputyDirectorOrDepEmpOrDeputyChiefOrClerk(

List<Staff> staffList) throws Exception {

System.out

.println("-----------------------------------------------------------------------------------------------");

System.out.println();

System.out.println("员工算分开始");

try {

int year = ConfigurationDataUtil.getIntegerVlaue("which\_year");

int month = ConfigurationDataUtil.getIntegerVlaue("which\_month");

// 获取任务进度

float task = new TaskProgressDao().getTaskProgress(year, month)

.getTaskProgress();

List<StaffScore> staffScoreList = new ArrayList<StaffScore>();

for (int z = 0; z < staffList.size(); z++) {

Staff staff = staffList.get(z);

System.out.println("员工名称：" + staff.getStaffName());

Map<Staff, List<StaffScoreRecord>> staffScoreRecordMap = new HashMap<Staff, List<StaffScoreRecord>>();

List<StaffScoreRecord> staffScoreRecordList = calculateScoreDao

.selectAllStaffScoreRecordByStaff(staff, year, month);

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

StaffScoreRecord staffScoreRecord = staffScoreRecordList

.get(i);

Staff fStaff = staffScoreRecord.getScoredFromStaff();

if (!staffScoreRecordMap.containsKey(fStaff)) {

List<StaffScoreRecord> staffScoredRecordFromList = new ArrayList<StaffScoreRecord>();

staffScoredRecordFromList.add(staffScoreRecord);

staffScoreRecordMap.put(fStaff,

staffScoredRecordFromList);

} else {

staffScoreRecordMap.get(fStaff).add(staffScoreRecord);

}

}

Map<Staff, Float> staffScoreMap = new HashMap<Staff, Float>();

Iterator iterator = staffScoreRecordMap.keySet().iterator();

String explain = "";// 打分说明

while (iterator.hasNext()) {

Staff staffKey = (Staff) iterator.next();

float totalScore = 0;

List<StaffScoreRecord> ssRecordList = staffScoreRecordMap

.get(staffKey);

// System.out.println(ssRecordList);

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

StaffScoreRecord record = ssRecordList.get(i);

totalScore = totalScore + record.getScore()

\* record.getStaffScoreItem().getScoreWeight();

System.out.println("打分记录:"

+ record.getScoredFromStaff().getStaffName()

+ ": " + record.getScore() + " \* "

+ record.getStaffScoreItem().getScoreWeight()

+ " totalScore是:" + totalScore);

}

staffScoreMap.put(staffKey, totalScore);

}

String postName = staff.getPost().getPostName();

String departmentName = staff.getDepartment()

.getDepartmentName();

float totalScore = 0;

if (postName.equals("市公司部门副主任")

&& departmentName.equals("人力资源部")) {

totalScore = calculateStaff(staff, staffScoreMap, year,

month, "总经理", "市公司部门主任", 0.3f, 0.7f, 0.5f, 0.5f);

} else if (postName.equals("市公司部门副主任")) {

totalScore = calculateStaff(staff, staffScoreMap, year,

month, "副总经理", "市公司部门主任", 0.3f, 0.7f, 0.5f, 0.5f);

} else if (postName.equals("市公司员工")) {

totalScore = calculateStaff(staff, staffScoreMap, year,

month, "市公司部门主任", "市公司部门副主任", 0.7f, 0.3f, 0.7f,

0.3f);

}

System.out.println("所属部门性质："

+ staff.getDepartment().getNature());

if (staff.getDepartment().getNature() == 0) {

float fuck = totalScore;

totalScore = task \* 0.7f + totalScore \* 0.3f;

System.out.println(staff.getStaffName() + "的最终得分: " + task

+ "\*0.7f+" + fuck + "\*0.3f = " + totalScore);

}

if (staff.getDepartment().getNature() == 1) {

float fuck = totalScore;

totalScore = task \* 0.5f + totalScore \* 0.5f;

System.out.println(staff.getStaffName() + "的最终得分: " + task

+ "\*0.5f+" + fuck + "\*0.5f = " + totalScore);

}

DecimalFormat df = new DecimalFormat("#.00");

totalScore = Float.parseFloat(df.format(totalScore));

StaffScore staffScore = new StaffScore();

staffScore.setStaff(staff);

staffScore.setTotalScore(totalScore);

staffScore.setYear(year);

staffScore.setMonth(month);

staffScore.setInstruction("");

staffScoreList.add(staffScore);

System.out.println("处理后为" + totalScore);

System.out.println();

}

System.out.println("员工算分结束");

return staffScoreList;

} catch (Exception e) {

e.printStackTrace();

return null;

}

}

private List<StaffScore> calculateTheScoreOfDeputyManager(

List<Staff> staffAList, List<StaffScore> staffScoreBList)

throws Exception {

System.out

.println("--------------------------------------------------------------");

System.out.println("分管副总算分开始：");

int year = ConfigurationDataUtil.getIntegerVlaue("which\_year");

int month = ConfigurationDataUtil.getIntegerVlaue("which\_month");

Map<Staff, List<StaffScore>> staffScoreMap = new HashMap<Staff, List<StaffScore>>();

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

Staff staff = staffAList.get(i);

List<StaffScore> staffScoreBs = new ArrayList<StaffScore>();

for (int j = staffScoreBList.size() - 1; j >= 0; j--) {

if (staff.getDepartment().getDepartmentId() == staffScoreBList

.get(j).getStaff().getDepartment()

.getHigherDepartmentId()) {

staffScoreBs.add(staffScoreBList.get(j));

}

}

staffScoreMap.put(staff, staffScoreBs);

}

List<StaffScore> staffAScoreList = new ArrayList<StaffScore>();

Iterator iterator = staffScoreMap.keySet().iterator();

while (iterator.hasNext()) {

Staff staffKey = (Staff) iterator.next();

System.out.println(staffKey.getStaffName());

System.out.println(" " + "分管部门数量：");

List<StaffScore> staffScoreBListByStaff = staffScoreMap

.get(staffKey);

System.out.println(staffScoreBListByStaff.size());

System.out.println(" " + "分管部门及得分：");

float score = 0f;

String explain = "";

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

System.out.println(staffScoreBListByStaff.get(i).getStaff()

.getDepartment().getDepartmentName()

+ "," + staffScoreBListByStaff.get(i).getTotalScore());

score = score + staffScoreBListByStaff.get(i).getTotalScore();

explain = explain

+ staffScoreBListByStaff.get(i).getInstruction() + ";";

}

score = score / staffScoreBListByStaff.size();

DecimalFormat df = new DecimalFormat("#.00");

score = Float.parseFloat(df.format(score));

System.out.println("最后得分：" + score);

StaffScore staffScore = new StaffScore();

staffScore.setStaff(staffKey);

staffScore.setTotalScore(score);

staffScore.setYear(year);

staffScore.setMonth(month);

staffScore.setInstruction(explain);

staffAScoreList.add(staffScore);

}

return staffAScoreList;

}

}

1. 部门个人模拟打分代码 部门个人模拟打分.sql：

DELIMITER @@

drop procedure if exists staffscore1;

create procedure staffscore1()

begin

declare stafffromid int;

declare depfromid int;

declare done int;

declare oldstafffromid int default 0;

declare cursor1 cursor for

select staff\_id, department\_id from staff where post\_id=2 or post\_id=1;

declare continue handler for not found set done=1;

open cursor1;

cursor\_loop:loop

if done=1 then

leave cursor\_loop;

end if;

fetch cursor1 into stafffromid, depfromid;

if oldstafffromid != stafffromid then

insert into staff\_score\_record(scored\_from\_staff\_id, scored\_to\_staff\_id, score\_item\_id, score, score\_date, which\_year, which\_month, modify\_times, owner)

select stafffromid, a.staff\_id, b.score\_item\_id, 99, '2015-06-30', 2015, 6, 3, stafffromid

from staff a, staff\_score\_relation b

where a.department\_id in (select department\_id from department where higher\_department\_id=depfromid)

and a.post\_id=4

and a.department\_id = b.department\_id

and a.post\_id = b.post\_id;

set oldstafffromid=stafffromid;

end if;

end loop cursor\_loop;

close cursor1;

end;

drop procedure if exists staffscore2;

create procedure staffscore2()

begin

declare stafffromid int;

declare depfromid int;

declare done int;

declare oldstafffromid int default 0;

declare cursor2 cursor for

select staff\_id, department\_id from staff where post\_id=3;

declare continue handler for not found set done=1;

open cursor2;

cursor\_loop:loop

if done=1 then

leave cursor\_loop;

end if;

fetch cursor2 into stafffromid, depfromid;

if oldstafffromid != stafffromid then

insert into staff\_score\_record(scored\_from\_staff\_id, scored\_to\_staff\_id, score\_item\_id, score, score\_date, which\_year, which\_month, modify\_times, owner)

select stafffromid, a.staff\_id, b.score\_item\_id, 99, '2015-06-30', 2015, 6, 3, stafffromid

from staff a, staff\_score\_relation b

where a.department\_id = depfromid

and (a.post\_id=4 or a.post\_id=5)

and a.department\_id = b.department\_id

and a.post\_id = b.post\_id;

set oldstafffromid=stafffromid;

end if;

end loop cursor\_loop;

close cursor2;

end;

drop procedure if exists staffscore3;

create procedure staffscore3()

begin

declare stafffromid int;

declare depfromid int;

declare done int;

declare oldstafffromid int default 0;

declare cursor3 cursor for

select staff\_id, department\_id from staff where post\_id=4;

declare continue handler for not found set done=1;

open cursor3;

cursor\_loop:loop

if done=1 then

leave cursor\_loop;

end if;

fetch cursor3 into stafffromid, depfromid;

if oldstafffromid != stafffromid then

insert into staff\_score\_record(scored\_from\_staff\_id, scored\_to\_staff\_id, score\_item\_id, score, score\_date, which\_year, which\_month, modify\_times, owner)

select stafffromid, a.staff\_id, b.score\_item\_id, 99, '2015-06-30', 2015, 6, 3, stafffromid

from staff a, staff\_score\_relation b

where a.department\_id = depfromid

and a.post\_id=5

and a.department\_id = b.department\_id

and a.post\_id = b.post\_id;

set oldstafffromid=stafffromid;

end if;

end loop cursor\_loop;

close cursor3;

end;

drop procedure if exists staffscore4;

create procedure staffscore4()

begin

declare stafffromid int;

declare depfromid int;

declare done int;

declare oldstafffromid int default 0;

declare cursor4 cursor for

select staff\_id, department\_id from staff where post\_id=7;

declare continue handler for not found set done=1;

open cursor4;

cursor\_loop:loop

if done=1 then

leave cursor\_loop;

end if;

fetch cursor4 into stafffromid, depfromid;

if oldstafffromid != stafffromid then

insert into staff\_score\_record(scored\_from\_staff\_id, scored\_to\_staff\_id, score\_item\_id, score, score\_date, which\_year, which\_month, modify\_times, owner)

select stafffromid, a.staff\_id, b.score\_item\_id, 99, '2015-06-30', 2015, 6, 3, stafffromid

from staff a, staff\_score\_relation b

where a.department\_id in (select department\_id from department where higher\_department\_id=depfromid)

and a.post\_id=9

and a.department\_id = b.department\_id

and a.post\_id = b.post\_id;

set oldstafffromid=stafffromid;

end if;

end loop cursor\_loop;

close cursor4;

end;

drop procedure if exists staffscore5;

create procedure staffscore5()

begin

declare stafffromid int;

declare depfromid int;

declare done int;

declare oldstafffromid int default 0;

declare cursor5 cursor for

select staff\_id, department\_id from staff where post\_id=8;

declare continue handler for not found set done=1;

open cursor5;

cursor\_loop:loop

if done=1 then

leave cursor\_loop;

end if;

fetch cursor5 into stafffromid, depfromid;

if oldstafffromid != stafffromid then

insert into staff\_score\_record(scored\_from\_staff\_id, scored\_to\_staff\_id, score\_item\_id, score, score\_date, which\_year, which\_month, modify\_times, owner)

select stafffromid, a.staff\_id, b.score\_item\_id, 99, '2015-06-30', 2015, 6, 3, stafffromid

from staff a, staff\_score\_relation b

where a.department\_id = depfromid

and (a.post\_id=9 or a.post\_id=10)

and a.department\_id = b.department\_id

and a.post\_id = b.post\_id;

set oldstafffromid=stafffromid;

end if;

end loop cursor\_loop;

close cursor5;

end;

drop procedure if exists staffscore6;

create procedure staffscore6()

begin

declare stafffromid int;

declare depfromid int;

declare done int;

declare oldstafffromid int default 0;

declare cursor6 cursor for

select staff\_id, department\_id from staff where post\_id=9;

declare continue handler for not found set done=1;

open cursor6;

cursor\_loop:loop

if done=1 then

leave cursor\_loop;

end if;

fetch cursor6 into stafffromid, depfromid;

if oldstafffromid != stafffromid then

insert into staff\_score\_record(scored\_from\_staff\_id, scored\_to\_staff\_id, score\_item\_id, score, score\_date, which\_year, which\_month, modify\_times, owner)

select stafffromid, a.staff\_id, b.score\_item\_id, 99, '2015-06-30', 2015, 6, 3, stafffromid

from staff a, staff\_score\_relation b

where a.department\_id = depfromid

and a.post\_id=10

and a.department\_id = b.department\_id

and a.post\_id = b.post\_id;

set oldstafffromid=stafffromid;

end if;

end loop cursor\_loop;

close cursor6;

end;

drop procedure if exists departmentscore2;

create procedure departmentscore2()

begin

declare staffid int;

declare depid int;

declare done int default 0;

declare depcomment text default 'hehe';

declare depdate date default '2015-06-30';

declare depyear int default 2015;

declare depmonth int default 10;

declare depmoditimes int default 3;

declare cursor6 cursor for

select staff\_id, department\_id from staff where role\_id in (select role\_id from role\_funcpage\_relation where funcpage\_id = 16);

declare continue handler for not found set done=1;

open cursor6;

repeat

fetch cursor6 into staffid, depid;

if done <> 1 then

-- do something

insert into department\_score\_record(department\_from\_id, department\_to\_id, score\_item\_id, score, score\_comment, score\_date, which\_year, which\_month, modify\_times, owner)

select a.department\_id, b.department\_id, a.score\_item\_id, b.score\_value, depcomment, depdate, depyear, depmonth, depmoditimes, staffid

from department\_score\_relation a, department\_score\_item b

where a.score\_item\_id = b.score\_item\_id

and a.department\_id = depid;

end if;

until done = 1 end repeat;

close cursor6;

end;

@@

DELIMITER ;

-- delete from department\_score\_record;

-- delete from staff\_score\_record;

-- delete from department\_score;

-- delete from staff\_score;

call departmentscore2();

call staffscore1();

call staffscore2();

call staffscore3();

-- call staffscore4();

-- call staffscore5();

-- call staffscore6();

update department\_score\_record set score\_date='2015-06-30', which\_year=2015, which\_month=6;

update staff\_score\_record set score\_date='2015-06-30', which\_year=2015, which\_month=6;

select 'over!!!';

1. Excel导入脚本中解析部门编号算法代码parseNumbers.py：

#-\*-coding:utf-8-\*-

import re

str = '分管副总、关联部门[3, 10-19]'

#/'fuck you' --> 'fuckyou'

def trim(str):

return ''.join(str.split(' '))

# '1-4' --> [1, 2, 3, 4]

def getNumbers(str):

#print 'geting number %s'%str

start, end = str.split('-')

return range(int(start), int(end)+1)

def parse(str):

patten = re.compile(r'\[[,\d-]\*\]')

numbersets = patten.findall(trim(str))

ret = []

for numberset in numbersets:

#print '-------------------begin------------------------------'

#print 'numberset:',numberset

numbers = numberset[1:-1].split(',')

#print 'numbers:',numbers

for num in numbers:

#print 'num:',num

if num.isdigit():

ret.append(int(num))

#print 'single number added : %s' % num

elif num.find('-') != -1:

ret.extend(getNumbers(num))

#print '-------------------end------------------------------'

#print

return ret

#print parse(str)

**三、取得的成就**

本系统现已正式上线运行，应用于本公司的每月绩效考评和例会以及学习培训资料上传，并进行了3次系统更新升级和服务器维护，已经成为了本公司的日常常用软件。