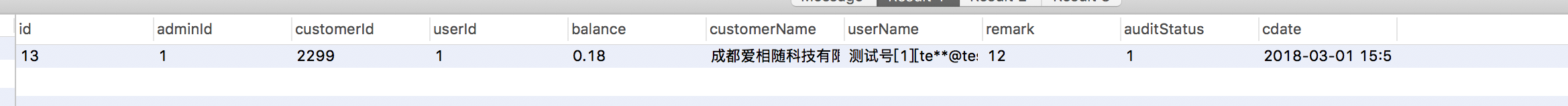
[在此处输入文章标题]

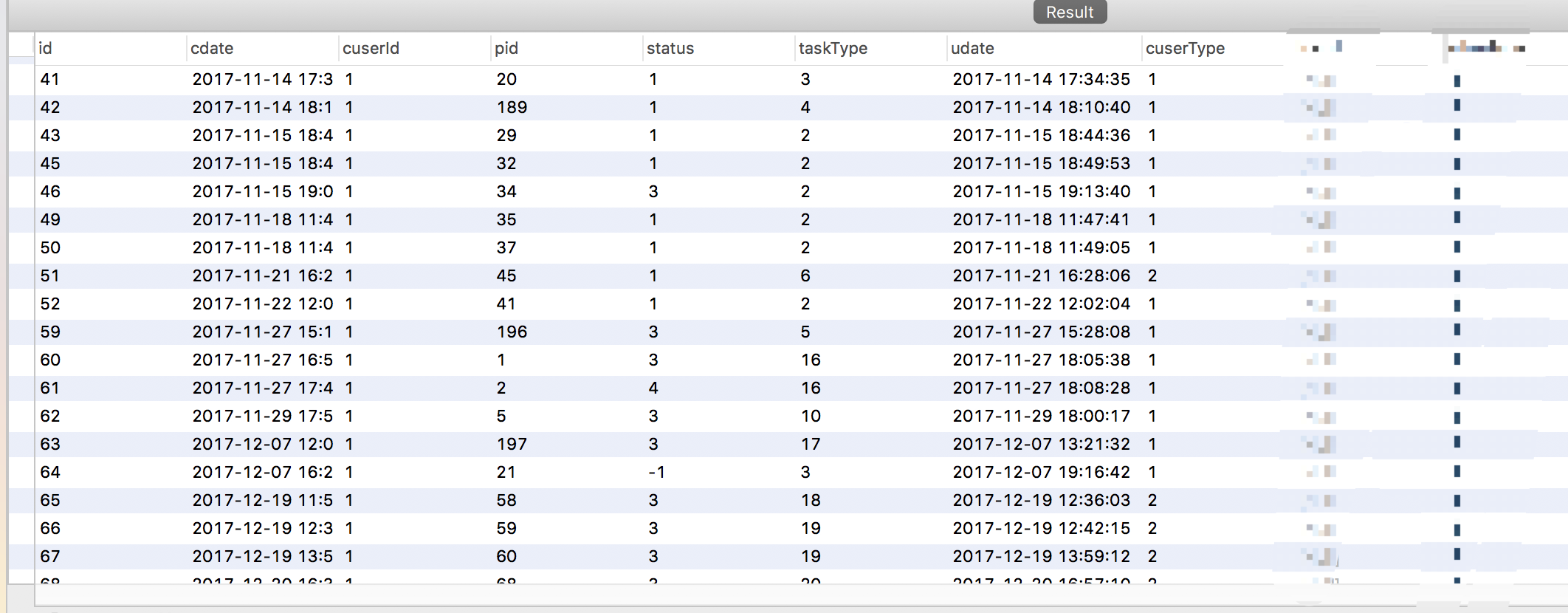
# 以老后台余额转移审核申请为例说明



# 1、审核基础信息

## 1、数据表 workflow\_task

@Entity  
@Table(name = "workflow\_task")  
public class WorkflowTask implements Serializable {  
  
 private static final long ***serialVersionUID*** = 4198682330135502141L;  
  
 public static final int ***SPECIALMONEYENDSHOWFLAG\_NO*** = 0;  
 public static final int ***SPECIALMONEYENDSHOWFLAG\_YES*** = 1;  
  
 @Id  
 @GeneratedValue(strategy = GenerationType.***IDENTITY***)  
 private Long id;  
  
 private Integer taskType; //EnumWorkflowTaskType 申请审核类型  
  
 private Long cuserId; //申请人   
  
 private Long pid; //申请类型对应的 具体申请记录的ID ，   
  
 private Integer status; //审批状态 EnumWorkflowTaskStatus 比如 审核中，审核同意，等  
  
 private Integer cuserType; //申请人类型 EnumWorkflowTaskCuserType //比如多点员工，多点管理人  
  
 @Temporal(TemporalType.***TIMESTAMP***)  
 @Column(columnDefinition = "TIMESTAMP DEFAULT CURRENT\_TIMESTAMP", insertable = true, updatable = false)  
 private Date cdate; //创建时间  
  
 @Temporal(TemporalType.***TIMESTAMP***)  
 @Column(columnDefinition = "TIMESTAMP DEFAULT CURRENT\_TIMESTAMP", insertable = true, updatable = true)  
 private Date udate; //最后处理时间  
  
 @Transient  
 private String taskTypeDesc;  
  
 @Transient  
 private String cuserName; //创建管理人员  
  
 @Transient  
 private String statusDesc; //审核状态描述  
  
 @Transient  
 private String remark\_1; //摘要，审核状态下进行预览  
 @Transient  
 private String remark\_2;  
 @Transient  
 private String remark\_3;  
  
 @Transient  
 private Integer specialMoneyEndShowFlag; //0-不显示结束 1-显示结束



## 2、枚举实体类

### 1、申请审核类型

public enum EnumWorkflowTaskType {  
  
 ***预开发票申请***(1,"预开发票申请"),  
 ***特批额度申请***(2,"特批额度申请"),  
 ***特批单价申请***(3,"特批单价申请"),  
 ***应用推广代金券申请***(4,"应用推广代金券申请"),  
 ***多点宝代金券申请***(5,"多点宝代金券申请"),  
 ***汇款底单审核***(6,"汇款底单审核"),  
 ***客户合同审核***(7,"客户合同审核"),  
 ***拜访客户审核***(8,"拜访客户审核"),  
 ***客户实名认证审核***(9,"客户实名认证审核"),  
 ***客户发票信息审核***(10,"客户发票信息审核"),

### 2、审核状态

\*/  
public enum EnumWorkflowTaskStatus {  
  
 ***等待审批***(1,"等待审批"),   
 ***审批中***(2,"审批中"),//当有多个人审核的时候，第一个人审核了，这个时候，状态就会变成审批中  
 ***审批完成\_同意***(3,"审批完成（同意）"),  
 ***审批完成\_拒绝***(4,"审批完成（已拒绝）"),  
 ***撤回作废***(-1,"撤回作废"),  
  
 ***未知***(99,"未知")  
 ;

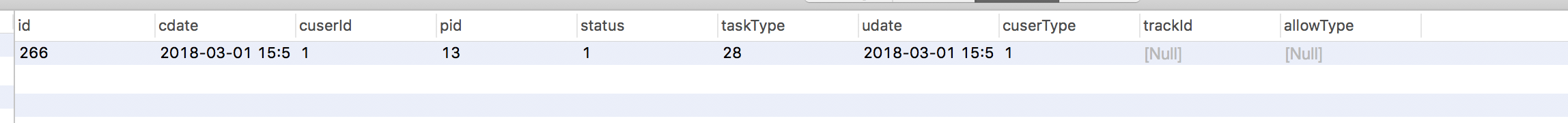
### 3、审批人类型 （一般用不上）

public enum EnumWorkflowTaskCuserType {  
  
 ***多点员工***(1,"多点员工"),  
 ***广告主***(2,"广告主"),  
  
 ***未知***(99,"未知")  
 ;

## 3、页面展示



## 4、数据库数据



# 3、申请人、审批人和抄送人集合 实体类

## 1、数据表 WorkflowTaskExamineDetails

@Entity  
@Table(name = "workflow\_task\_examine\_details",  
 indexes = {  
 @Index(name = "task\_adm\_uniq", columnList = "taskId,admId", unique = true)  
 }  
)  
public class WorkflowTaskExamineDetails implements Serializable {  
  
 private static final long ***serialVersionUID*** = 4198682330135502141L;  
  
 @Id  
 @GeneratedValue(strategy = GenerationType.***IDENTITY***)  
 private Long id;  
  
 private Long taskId; //WorkflowTask id  
  
 private Long admId; //申请人 审批人 抄送人  
  
 private Integer admType; //EnumWorkflowTaskUserType，申请人 审批人 抄送人  
  
  
 private Integer examineOrder; //处理顺序 多个人审批时，多人共同和审核同一件事情，其数据是在创建审核信息的时候创建的  
  
 private Integer examineStatus; //当前处理状态 EnumWorkflowTaskExamineStatus  
  
 private String remark; //审批人审批时 批注、回复、说明  
  
 @Temporal(TemporalType.***TIMESTAMP***)  
 @Column(columnDefinition = "TIMESTAMP DEFAULT CURRENT\_TIMESTAMP", insertable = true, updatable = false)  
 private Date cdate; //创建时间  
  
 @Temporal(TemporalType.***TIMESTAMP***)  
 @Column(columnDefinition = "TIMESTAMP DEFAULT CURRENT\_TIMESTAMP", insertable = true, updatable = true)  
 private Date udate; //最后处理时间  
  
  
 @Transient  
 private String admName;  
  
 @Transient  
 private String examineStatusDesc;  
  
 @Transient  
 private Boolean currAdmUserFlag = false; //是否是当前登录人  
  
 @Transient  
 private Boolean currNodeFlag = false; //流程是否走到该节点 是否轮到该记录处理了  
  
 @Transient  
 private Boolean apprDetailsShowFlag = true; //后面等待审批的审批人  
  
  
 public WorkflowTaskExamineDetails() {  
 }  
  
 public WorkflowTaskExamineDetails(Long taskId, Long admId, Integer admType, Integer examineOrder, Integer examineStatus, String remark) {  
 this.taskId = taskId;  
 this.admId = admId;  
 this.admType = admType;  
 this.examineOrder = examineOrder;  
 this.examineStatus = examineStatus;  
 this.remark = remark;  
 }

## 2、枚举实体类

### 1、提交的审核类型

public enum EnumWorkflowTaskExamineStatus {  
  
 ***撤回作废***(-1,"撤回作废"),  
 ***发起申请***(0,"发起申请"),  
 ***等待审批***(1,"等待审批"),  
 ***审批通过***(2,"审批通过"),  
 ***审批拒绝***(3,"审批拒绝"),  
  
 ***未知***(99,"未知")  
 ;  
  
 public Integer type;  
 public String des;  
  
 EnumWorkflowTaskExamineStatus(Integer type, String des) {  
 this.type = type;  
 this.des = des;  
 }

### 2、人的类型

public enum EnumWorkflowTaskUserType {  
  
 ***申请人***(1,"申请人"),  
 ***审批人***(2,"审批人"),  
 ***抄送人***(3,"抄送人"),  
  
 ***未知***(99,"未知")  
 ;  
  
 public Integer type;  
 public String des;  
  
 EnumWorkflowTaskUserType(Integer type, String des) {  
 this.type = type;  
 this.des = des;  
 }

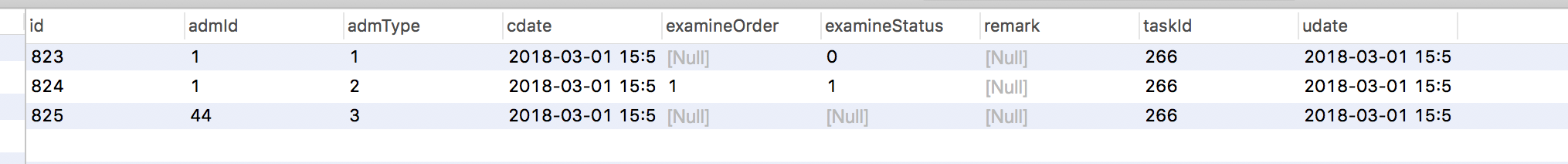
## 3、数据库数据

### 解释：admid 为1 为test账户 [44为yujin.zhang@duodian.com](mailto:44为yujin.zhang@duodian.com) 账户为抄送人

### 1、申请人为 adminId 第一条记录 ，审核人admid为第二条记录 ，[抄送人为yujin.zhang@duodian.com](mailto:抄送人为yujin.zhang@duodian.com) （admType 进行区分）

### 2、examineOrder 默认顺序是1 ，有多个审核人依次排下去。

### 3、examineStatus 为审核状态，所以当提交审核以后 ，先只有申请人和审核人有状态。



# 4、默认审核人

## 1、数据库表

### 1、 workflow\_task\_default，审核人是否可以修改

@Entity  
@Table(name = "workflow\_task\_default",  
 indexes = {  
 @Index(name = "type\_uniq", columnList = "taskType", unique = true)  
 }  
 )  
public class WorkflowTaskDefault implements Serializable {  
  
 private static final long ***serialVersionUID*** = 4198682330135502141L;  
  
 @Id  
 @GeneratedValue(strategy = GenerationType.***IDENTITY***)  
 private Long id;  
  
 private Integer taskType; //EnumWorkflowTaskType  
  
 private Long lastAdmId; //最后操作人  
  
 private Integer status; //状态 0-禁用 1-可用 EnumYesNo  
  
 private Integer modifyFlag; //申请时是否可修改审批人及抄送人（可以增加其他审批人或抄送人但是不能去掉默认的） 0-不可修改 1-可修改 EnumYesNo  
  
 @Temporal(TemporalType.***TIMESTAMP***)  
 @Column(columnDefinition = "TIMESTAMP DEFAULT CURRENT\_TIMESTAMP", insertable = true, updatable = false)  
 private Date cdate; //创建时间  
  
 @Temporal(TemporalType.***TIMESTAMP***)  
 @Column(columnDefinition = "TIMESTAMP DEFAULT CURRENT\_TIMESTAMP", insertable = true, updatable = true)  
 private Date udate; //最后处理时间  
  
 @Transient  
 private String taskTypeDesc;  
  
 @Transient  
 private String lastAdmName;  
  
 @Transient  
 private String statusDesc;  
  
 @Transient  
 private String modifyFlagDesc;  
  
 @Transient  
 private List<SysAdminUser> apprList;  
 @Transient  
 private List<SysAdminUser> ccList;  
  
 public WorkflowTaskDefault() {  
 }

### 2、workflow\_task\_default\_appr\_cc，默认审核人数据表

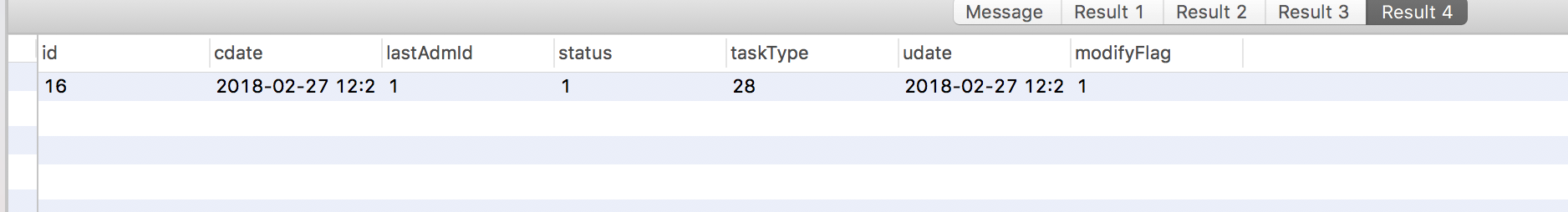
@Entity  
@Table(name = "workflow\_task\_default\_appr\_cc",  
 indexes = {  
 @Index(name = "type\_adm\_uniq", columnList = "taskType,admId", unique = true)  
 }  
)  
public class WorkflowTaskDefaultApprCc implements Serializable {  
  
 private static final long ***serialVersionUID*** = 4198682330135502141L;  
  
 @Id  
 @GeneratedValue(strategy = GenerationType.***IDENTITY***)  
 private Long id;  
  
 private Integer taskType; //EnumWorkflowTaskType  
  
 private Long admId; //后台用户ID  
  
 private Integer admType; //EnumWorkflowTaskUserType 审批人还是抄送人  
  
 private Integer examineOrder; //处理顺序 多个人审批时（当创建默认审批人时候创建的，）  
  
  
 public WorkflowTaskDefaultApprCc() {  
 }

## 2、数据库数据

### 1、workflow\_task\_default

# 关于审核类型为 是否可用和是否可以修改审核人 。抄送人

#### select \* from workflo\_task\_default w where w.taskType ='28' order by w.cdate desc;



### 2、workflow\_task\_default\_appr\_cc

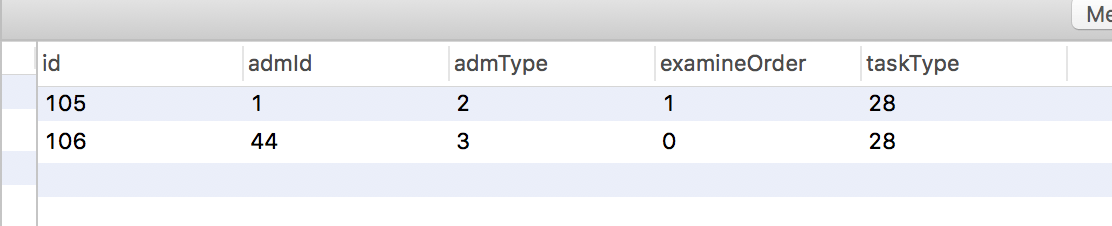


#### select \* from workflow\_task\_default\_appr\_cc w where w.taskType = '28';

#### 解释：

##### 1、第一条为审核人记录 examineOrder为审核顺序

##### 2、第二条为抄送人数据 examineOrder 为0表示不需要顺序



## 3、代码逻辑（保存审核信息的时候的代码）

### 1、判断是修改了审核人，判断是否可以审核人，和获取审核人列表进行

WorkflowTaskDefault taskDefault = workflowTaskDefaultDAO.findTopByTaskType(taskType);  
if (taskDefault != null && taskDefault.getModifyFlag() != null && taskDefault.getModifyFlag().compareTo(EnumYesNo.***NO***.status) == 0){  
 //不可修改默认设置 （可增加人 不可去掉默认的人）  
 List<SysAdminUser> apprList = workflowMapper.findWorkflowTaskDefaultAdmListByTaskTypeAndAdmType(taskType,EnumWorkflowTaskUserType.***审批人***.type);  
 if (CollectionUtils.isNotEmpty(apprList)){  
 for (SysAdminUser apprUser : apprList){  
 Boolean flag = false;  
 for (String apprStr : apprStrList){  
 if (apprUser.getId().compareTo(Long.valueOf(apprStr)) == 0){  
 flag = true;  
 break;  
 }  
 }  
 if (!flag){  
 //没有这个人  
 throw new AppException("该申请不可去掉默认的审批人及抄送人");  
 }  
 }  
 }  
}

## 2、取得管理员用户，从审核人，抄送人表中 ，提取 类型admType 是审批人的，并且审批人配置表中的状态是可用的

<select id="findWorkflowTaskDefaultAdmListByTaskTypeAndAdmType" resultType="com.duodian.admore.entity.db.admin.SysAdminUser">  
 SELECT a.*\** FROM workflow\_task\_default\_appr\_cc t  
 LEFT JOIN sys\_admin\_user a ON a.id = t.admId  
 LEFT JOIN workflow\_task\_default b ON b.taskType = t.taskType  
 WHERE t.taskType = #{taskType} AND t.admType = #{admType} AND b.status = 1  
 ORDER BY t.examineOrder ASC , t.admId ASC ;  
</select>

# 5、发起一个审核信息，进入数据库中存储（去掉了钉钉发消息）

public void saveWorkflowTaskAndDetailsByAllowType(Integer taskType, Long cuserId, String Long pid, String apprIds, String ccIds) {  
 WorkflowTask workflowTask = null;  
 workflowTask = new WorkflowTask(taskType, cuserId,

EnumWorkflowTaskCuserType.***多点员工***.type, pid, EnumWorkflowTaskStatus.***等待审批***.type);  
 //处理工作流相关,预览页面查看详情  
 workflowTaskDAO.save(workflowTask);  
  
 //申请人、审核人、抄送人 详细信息。先放入申请人信息  
 WorkflowTaskExamineDetails cuserDetails = new WorkflowTaskExamineDetails(workflowTask.getId(), cuserId,

EnumWorkflowTaskUserType.***申请人***.type, null,

EnumWorkflowTaskExamineStatus.***发起申请***.type, null);  
 workflowTaskExamineDetailsDAO.save(cuserDetails);  
  
 List<String> defApprStrList = Arrays.asList(apprIds.split(","));  
 List<String> apprStrList = new ArrayList<>(defApprStrList);  
  
# 审核人和配置信息，观察是否可以修改审核人和抄送人  
 WorkflowTaskDefault taskDefault = workflowTaskDefaultDAO.findTopByTaskType(taskType);  
 if (taskDefault != null && taskDefault.getModifyFlag() != null &&

taskDefault.getModifyFlag().compareTo(EnumYesNo.***NO***.status) == 0){  
 //查找关于某一审核类型的审核人信息。（其中还有关于审核人配置是否可用等等，上面有介绍）   
 List<SysAdminUser> apprList = workflowMapper.findWorkflowTaskDefaultAdmListByTaskTypeAndAdmType(

taskType,EnumWorkflowTaskUserType.***审批人***.type);  
 if (CollectionUtils.isNotEmpty(apprList)){  
 for (SysAdminUser apprUser : apprList){  
 Boolean flag = false;  
 for (String apprStr : apprStrList){  
 if (apprUser.getId().compareTo(Long.valueOf(apprStr)) == 0){  
 flag = true;  
 break;  
 }  
 }  
 if (!flag){  
 //没有这个人  
 throw new AppException("该申请不可去掉默认的审批人及抄送人");  
 }  
 }  
 }  
 }  
 Long admId = null;  
 String content = null;  
 int i = 1;  
  
 List<String> existApprList = new ArrayList<>();  
 List<String> existCcList = new ArrayList<>();  
  
 for (String appsStr : apprStrList) {  
 if (existApprList.contains(appsStr)){  
 continue;  
 }

#给审核人添加数据，并配置审核人审核的顺序  
 WorkflowTaskExamineDetails appDetails = new WorkflowTaskExamineDetails(

workflowTask.getId(),

Long.parseLong(appsStr),

EnumWorkflowTaskUserType.***审批人***.type, i,

EnumWorkflowTaskExamineStatus.***等待审批***.type, null);  
 workflowTaskExamineDetailsDAO.save(appDetails);  
 existApprList.add(appsStr);  
 i++;  
 }  
 #检查看看有没有抄送人，已经抄送人是是否已经修改了，和上面的审核人其实是一样的  
 if (StringUtils.isNotBlank(ccIds)) {  
 List<String> ccStrList = Arrays.asList(ccIds.split(","));  
 //检查默认设置  
 if (taskDefault != null && taskDefault.getModifyFlag() != null &&

taskDefault.getModifyFlag().compareTo(EnumYesNo.***NO***.status) == 0){  
 List<SysAdminUser> ccList = workflowMapper.findWorkflowTaskDefaultAdmListByTaskTypeAndAdmType(

taskType,EnumWorkflowTaskUserType.***抄送人***.type);  
 if (CollectionUtils.isNotEmpty(ccList)){  
 for (SysAdminUser ccUser : ccList){  
 Boolean flag = false;  
 for (String ccStr : ccStrList){  
 if (ccUser.getId().compareTo(Long.valueOf(ccStr)) == 0){  
 flag = true;  
 break;  
 }  
 }  
 if (!flag){  
 //没有这个人  
 throw new AppException("该申请不可去掉默认的审批人及抄送人");  
 }  
 }  
 }  
 }  
 for (String ccStr : ccStrList) {  
 if (existCcList.contains(ccStr)){  
 continue;  
 }  
 WorkflowTaskExamineDetails appDetails = new WorkflowTaskExamineDetails(workflowTask.getId(),

Long.parseLong(ccStr), EnumWorkflowTaskUserType.***抄送人***.type,

null, null, null);  
 workflowTaskExamineDetailsDAO.save(appDetails);  
  
 existCcList.add(ccStr);  
 }  
 } else {//如果为空，看看数据库中有没有抄送人  
 if (taskDefault != null && taskDefault.getModifyFlag() != null &&

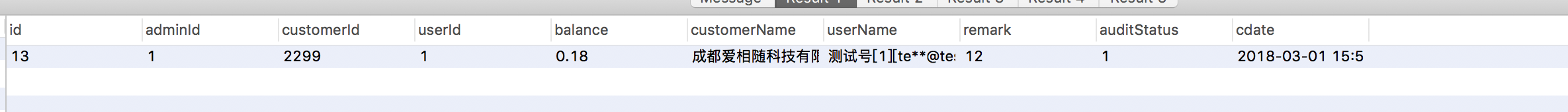
taskDefault.getModifyFlag().compareTo(EnumYesNo.***NO***.status) == 0){  
 List<SysAdminUser> ccList = workflowMapper.findWorkflowTaskDefaultAdmListByTaskTypeAndAdmType(

taskType,EnumWorkflowTaskUserType.***抄送人***.type);  
 if (CollectionUtils.isNotEmpty(ccList)){  
 throw new AppException("该申请不可去掉默认的审批人及抄送人");  
 }  
 }  
 }  
 if (admId != null) {  
 sendDingMsg(admId,content,workflowTask.getId());  
 }  
 }

### 2、数据库执行顺序

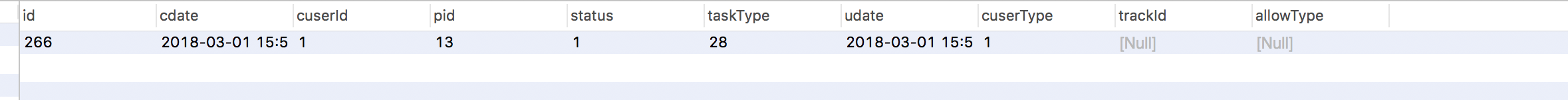
### 1、申请信息

#### select \* from user\_old\_background\_balance\_trannsfer u where u.id='13' order by cdate desc ;



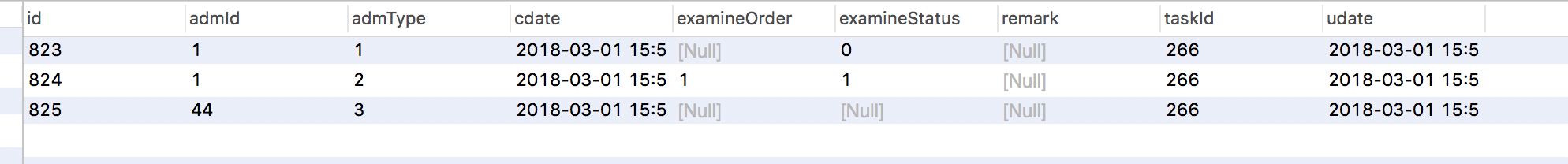
### 2、审核基础信息

#### select \* from workflow\_task w where w.pid = '13' and w.taskType ='28' order by w.cdate desc; # 出现一条记录,主键为266



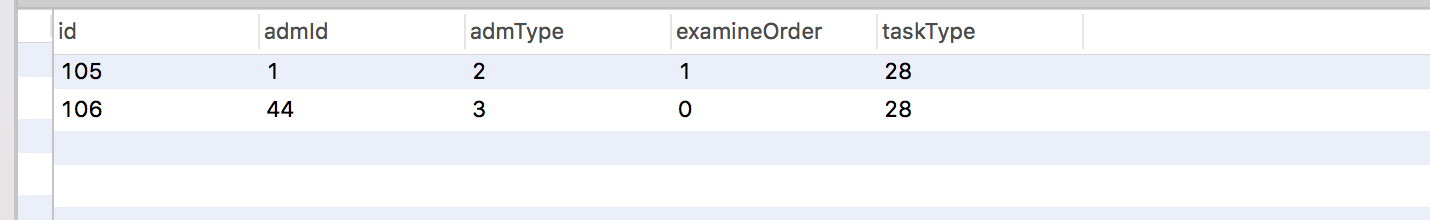
### 3、申请人，审批人，抄送人详情

#### select \* from workflow\_task\_examine\_details w where w.taskId = '266' ;# 出现三条记录



### 4、审批人和抄送人数据库

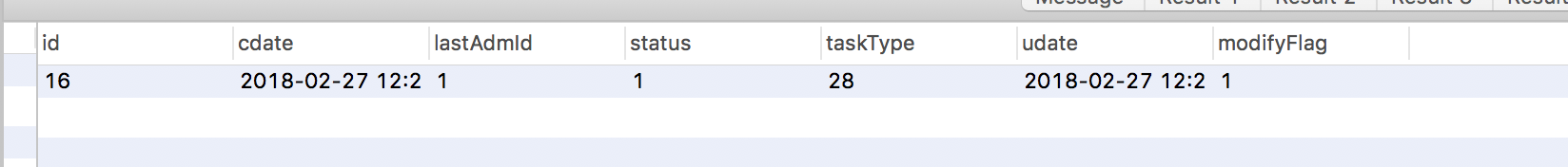
#### select \* from workflow\_task\_default\_appr\_cc w where w.taskType = '28'; #出现2条记录 审批人 抄送人数据



### 5、审核人，审批人配置可否修改和可用

#### # 关于审核类型为 是否可用和是否可以修改审核人

#### select \* from workflow\_task\_default w where w.taskType ='28' order by w.cdate desc;



# 6、未读条数（待我我审批和抄送我的）

## 1、未读条实体

public class WorkflowUnreadBadgeData implements Serializable{  
  
 private static final long ***serialVersionUID*** = -1211726511402154326L;  
  
 private Integer waitUnreadCount; //待我审批的 未读条数  
 private Integer copyUnreadCount; //抄送我的 未读条数  
  
  
 public WorkflowUnreadBadgeData() {  
 }

## 1、controller

@RequestMapping("getUnreadBadge")  
@ResponseBody  
public ResponseBean getUnreadBadge() {  
 try {  
 SysAdminUser adminUser = AppSessionHelper.getSessionUser();  
 return ResponseBean.buildSuccess(workflowService.getUnreadBadgeByAdmId( adminUser.getId()));  
 } catch (AppException e) {  
 return ResponseBean.buildFailure(e.getCode(), e.getMessage());  
 } catch (Exception e) {  
 logger.error(e.getMessage(), e);  
 return ResponseBean.buildFailure(ErrorCodeEnum.***系统错误***);  
 }  
}

## 2、service

@Override  
public WorkflowUnreadBadgeData getUnreadBadgeByAdmId(Long admId) {  
  
 WorkflowQuery query = new WorkflowQuery();  
  
 List<Integer> examineStatusList = new ArrayList<>();  
 examineStatusList.add(EnumWorkflowTaskExamineStatus.***等待审批***.type);  
  
 List<Integer> taskStatusList = new ArrayList<>();  
 taskStatusList.add(EnumWorkflowTaskStatus.***等待审批***.type); workflow\_task  
 taskStatusList.add(EnumWorkflowTaskStatus.***审批中***.type);  
  
 query.setAdmId(admId);  
 query.setAdmType(EnumWorkflowTaskUserType.***审批人***.type);  
 query.setPageType(EnumWorkflowPageType.***待我审批***.type);  
 query.setExamineStatusList(examineStatusList);  
 query.setTaskStatusList(taskStatusList);  
  
 Long waitCount = workflowMapper.countWorkflowPageData(query);  
 Long copyCount = workflowTaskUnreadBadgeDAO.countByAdmIdAndPageType(admId, EnumWorkflowPageType.***抄送我的***.type);  
  
 return new WorkflowUnreadBadgeData(waitCount == null ? 0 : waitCount.intValue(), copyCount == null ? 0 : copyCount.intValue());  
}

## 3、mybatis （待审核事多，为什么呢）

<select id="countWorkflowPageData" resultType="java.lang.Long">  
 SELECT count(DISTINCT t.id) FROM workflow\_task t  
 LEFT JOIN workflow\_task\_examine\_details a ON a.taskId = t.id  
 LEFT JOIN sys\_admin\_user b ON b.id = t.cuserId AND t.cuserType = 1 //确定基本的信息匹配  
 <if test="pageType != null">  
 <choose>

有时候审批人不止一个，需要筛选掉，还轮不到他审批的人  
 <when test="pageType == 1"> //1代表是  
 LEFT JOIN (  
 SELECT c2.*\** FROM workflow\_task\_examine\_details c2  
 LEFT JOIN (  
 SELECT c1.taskId,min(c1.id) minId FROM workflow\_task\_examine\_details c1  
 WHERE c1.admType = 2 AND c1.examineStatus = 1 GROUP BY c1.taskId // admType为2表示待审核，其中包括审核人、审批人和抄送人，所以用了分组  
 ) c3 ON c3.minId = c2.id  
 WHERE c3.taskId > 0  
 ) c ON c.taskId = t.id  
 </when>  
 </choose>  
 </if>  
 <where>  
 <include refid="getWorkflowPageDataWhere"></include>  
 </where>  
</select>  
  
<sql id="getWorkflowPageDataWhere">  
 <if test="taskType != null">  
 AND t.taskType = #{taskType}  
 </if>  
 <if test="startDate != null">  
 AND t.cdate &gt;= #{startDate}  
 </if>  
 <if test="endDate != null">  
 AND t.cdate &lt;= #{endDate}  
 </if>  
 <if test="status != null">  
 AND t.status = #{status}  
 </if>  
 <if test="taskStatusList != null and taskStatusList.size() > 0">  
 AND t.status IN  
 <foreach collection="taskStatusList" index="index" item="ts" open="(" separator="," close=")">  
 #{ts}  
 </foreach>  
 </if>  
 <if test="cuserParam != null and cuserParam != ''">  
 AND (b.name LIKE CONCAT('%',#{cuserParam},'%' ) OR b.email LIKE CONCAT('%',#{cuserParam},'%' ))  
 </if>  
 <if test="admId != null">  
 AND a.admId = #{admId}  
 </if>  
 <if test="admType != null">  
 AND a.admType = #{admType}  
 </if>  
 <if test="examineStatusList != null and examineStatusList.size() > 0">  
 AND a.examineStatus IN  
 <foreach collection="examineStatusList" index="index" item="es" open="(" separator="," close=")">  
 #{es}  
 </foreach>  
 </if>  
 <if test="pageType != null">  
 <choose>  
 <when test="pageType == 1">  
 AND c.admId = #{admId}  
 </when>  
 </choose>  
 </if>  
</sql>

## 4、枚举 查看下信息（一般也没什么用，只在待我审批中其实有用到，抄送人中代码中有写，但是没毛用的）

public enum EnumWorkflowPageType {  
  
 ***发起审批***(0,"发起审批"),  
 ***待我审批***(1,"待我审批"),  
 ***我已审批***(2,"我已审批"),  
 ***我发起的***(3,"我发起的"),  
 ***抄送我的***(4,"抄送我的"),  
  
 ***未知***(99,"未知")  
 ;  
  
 public Integer type;  
 public String des;  
  
 EnumWorkflowPageType(Integer type, String des) {  
 this.type = type;  
 this.des = des;  
 }

## 5、抄送人是在审核通过以后出现的，这里是单独做了一个实体类

@Entity  
@Table(name = "workflow\_task\_unread\_badge",  
 indexes = {  
 @Index(name = "i\_admId", columnList = "admId")  
 }  
)  
public class WorkflowTaskUnreadBadge implements Serializable {  
  
 private static final long ***serialVersionUID*** = 4198682330135502141L;  
  
 @Id  
 @GeneratedValue(strategy = GenerationType.***IDENTITY***)  
 private Long id;  
  
 private Long taskId;  
  
 private Long admId;  
  
 private Integer pageType; //EnumWorkflowPageType  
  
  
 public WorkflowTaskUnreadBadge() {  
 }

# 7、查询待我审批（有时候审批人不止一个，需要筛选掉，还轮不到他审批的人）

@RequestMapping("wait/data")  
@ResponseBody  
public ResponseBean data(WorkflowQuery query, @RequestParam(defaultValue = "0") Integer page, @RequestParam(defaultValue = "7") Integer pageSize) {  
 try {  
 SysAdminUser adminUser = AppSessionHelper.getSessionUser();  
 Pageable pageable = new PageRequest(page, pageSize);  
  
 List<Integer> examineStatusList = new ArrayList<>();  
 examineStatusList.add(EnumWorkflowTaskExamineStatus.***等待审批***.type); //审核人状态为等待审批  
  
  
 List<Integer> taskStatusList = new ArrayList<>();  
 taskStatusList.add(EnumWorkflowTaskStatus.***等待审批***.type);

taskStatusList.add(EnumWorkflowTaskStatus.***审批中***.type);  
  
 query.setAdmId(adminUser.getId());  
 query.setAdmType(EnumWorkflowTaskUserType.***审批人***.type);  
 query.setPageType(EnumWorkflowPageType.***待我审批***.type);  
 query.setExamineStatusList(examineStatusList);  
 query.setTaskStatusList(taskStatusList);  
  
 //删除未读标记  
 workflowService.deleteUnreadBadgeByAdmIdAndPageType(adminUser.getId(),EnumWorkflowPageType.***待我审批***.type);  
  
 return ResponseBean.buildSuccess(workflowService.getWorkflowPageData(pageable, query, adminUser.getId()));  
 } catch (AppException e) {  
 return ResponseBean.buildFailure(e.getCode(), e.getMessage());  
 } catch (Exception e) {  
 logger.error(e.getMessage(), e);  
 return ResponseBean.buildFailure(ErrorCodeEnum.***系统错误***);  
 }  
}

@Override  
public Page<WorkflowTask> getWorkflowPageData(Pageable pageable, WorkflowQuery query, Long admId) {  
 query.setStartDate(DateHelper.getDateFirstTime(query.getStartDate()));  
 query.setEndDate(DateHelper.getDateLastTime(query.getEndDate()));  
 query.setOffset(pageable.getOffset());  
 query.setLimit(pageable.getPageSize());  
  
 List<WorkflowTask> list = workflowMapper.getWorkflowPageDataList(query);  
 if (CollectionUtils.isNotEmpty(list)) {  
 for (WorkflowTask task : list) {  
 this.assembleTaskRemark(task);  
 }  
 }  
 Long count = workflowMapper.countWorkflowPageData(query); //和上面的未读条数是一样的  
  
 return new PageImpl<>(list, pageable, count);  
}

<select id="getWorkflowPageDataList" resultType="com.duodian.admore.entity.db.workflow.WorkflowTask">  
 SELECT DISTINCT t.*\**,ifnull(b.name,c.nickName) cuserName FROM workflow\_task t  
 LEFT JOIN workflow\_task\_examine\_details a ON a.taskId = t.id  
 LEFT JOIN sys\_admin\_user b ON b.id = t.cuserId AND t.cuserType = 1  
 LEFT JOIN user c ON c.id = t.cuserId AND t.cuserType = 2  
 <if test="pageType != null">  
 <choose>  
 <when test="pageType == 1">  
 LEFT JOIN (  
 SELECT c2.*\** FROM workflow\_task\_examine\_details c2  
 LEFT JOIN (  
 SELECT c1.taskId,min(c1.id) minId FROM workflow\_task\_examine\_details c1  
 WHERE c1.admType = 2 AND c1.examineStatus = 1 GROUP BY c1.taskId  
 ) c3 ON c3.minId = c2.id  
 WHERE c3.taskId > 0  
 ) c ON c.taskId = t.id  
 </when>  
 </choose>  
 </if>  
 <where>  
 <include refid="getWorkflowPageDataWhere"></include>  
 </where>  
 ORDER BY t.udate DESC  
 <if test="offset != null and limit != null">  
 limit #{offset}, #{limit}  
 </if>  
</select>

# 8、我已审批的

@RequestMapping("finish/data")  
@ResponseBody  
public ResponseBean finishata(WorkflowQuery query, @RequestParam(defaultValue = "0") Integer page, @RequestParam(defaultValue = "7") Integer pageSize) {  
 try {  
 SysAdminUser adminUser = AppSessionHelper.getSessionUser();  
 Pageable pageable = new PageRequest(page, pageSize);  
  
 List<Integer> examineStatusList = new ArrayList<>();  
 examineStatusList.add(EnumWorkflowTaskExamineStatus.***审批通过***.type);  
 examineStatusList.add(EnumWorkflowTaskExamineStatus.***审批拒绝***.type);  
  
 query.setAdmId(adminUser.getId());  
 query.setAdmType(EnumWorkflowTaskUserType.***审批人***.type);  
 query.setExamineStatusList(examineStatusList);  
  
 return ResponseBean.buildSuccess(workflowService.getWorkflowPageData(pageable, query, adminUser.getId()));  
 } catch (AppException e) {  
 return ResponseBean.buildFailure(e.getCode(), e.getMessage());  
 } catch (Exception e) {  
 logger.error(e.getMessage(), e);  
 return ResponseBean.buildFailure(ErrorCodeEnum.***系统错误***);  
 }  
}

# 9、我发起的（这个时候终于用到 detail中的申请人了）

@RequestMapping("start/data")  
@ResponseBody  
public ResponseBean startData(WorkflowQuery query, @RequestParam(defaultValue = "0") Integer page, @RequestParam(defaultValue = "7") Integer pageSize) {  
 try {  
 SysAdminUser adminUser = AppSessionHelper.getSessionUser();  
 Pageable pageable = new PageRequest(page, pageSize);  
  
 query.setAdmId(adminUser.getId());  
 query.setAdmType(EnumWorkflowTaskUserType.***申请人***.type);  
  
 return ResponseBean.buildSuccess(workflowService.getWorkflowPageData(pageable, query, adminUser.getId()));  
 } catch (AppException e) {  
 return ResponseBean.buildFailure(e.getCode(), e.getMessage());  
 } catch (Exception e) {  
 logger.error(e.getMessage(), e);  
 return ResponseBean.buildFailure(ErrorCodeEnum.***系统错误***);  
 }  
}

# 10、抄送我的

@RequestMapping("copy/data")  
 @ResponseBody  
 public ResponseBean copyData(WorkflowQuery query, @RequestParam(defaultValue = "0") Integer page, @RequestParam(defaultValue = "7") Integer pageSize) {  
 try {  
 SysAdminUser adminUser = AppSessionHelper.getSessionUser();  
 Pageable pageable = new PageRequest(page, pageSize);  
  
 query.setAdmId(adminUser.getId());  
 query.setAdmType(EnumWorkflowTaskUserType.***抄送人***.type);  
  
 List<Integer> taskStatusList = new ArrayList<>();  
 taskStatusList.add(EnumWorkflowTaskStatus.***审批完成\_同意***.type);  
// taskStatusList.add(EnumWorkflowTaskStatus.审批完成\_拒绝.type); //拒绝的 不需要抄送  
  
 query.setTaskStatusList(taskStatusList);  
 query.setPageType(EnumWorkflowPageType.***抄送我的***.type);  
  
 //删除未读标记  
 workflowService.deleteUnreadBadgeByAdmIdAndPageType(adminUser.getId(),EnumWorkflowPageType.***抄送我的***.type);  
  
 return ResponseBean.buildSuccess(workflowService.getWorkflowPageData(pageable, query, adminUser.getId()));  
 } catch (AppException e) {  
 return ResponseBean.buildFailure(e.getCode(), e.getMessage());  
 } catch (Exception e) {  
 logger.error(e.getMessage(), e);  
 return ResponseBean.buildFailure(ErrorCodeEnum.***系统错误***);  
 }  
 }

# 11、获取审核详情

@RequestMapping("getDetailsDataByTaskId")  
@ResponseBody  
public ResponseBean getDetailsDataByTaskId(Long taskId) {  
 try {  
 SysAdminUser adminUser = AppSessionHelper.getSessionUser();  
 return ResponseBean.buildSuccess(workflowService.getDetailsDataByTaskId(taskId, adminUser.getId()));  
 } catch (AppException e) {  
 return ResponseBean.buildFailure(e.getCode(), e.getMessage());  
 } catch (Exception e) {  
 logger.error(e.getMessage(), e);  
 return ResponseBean.buildFailure(ErrorCodeEnum.***系统错误***);  
 }  
}

@Override  
public WorkflowDetailsData getDetailsDataByTaskId(Long taskId, Long admId) {  
  
 if (taskId == null || taskId <= 0) {  
 throw new AppException("没有获取到记录ID，请刷新页面重试！");  
 }  
  
 Long count = workflowTaskExamineDetailsDAO.countByTaskIdAndAdmId(taskId, admId);  
 if (count == null || count == 0) {  
 throw new AppException("没有查询到您要查看的记录详情[id:" + taskId + "]");  
 }  
 WorkflowTask workflowTask = workflowTaskDAO.findOne(taskId);  
 if (workflowTask != null) {  
 WorkflowDetailsData detailsData = new WorkflowDetailsData();  
 List<WorkflowLabelValueBean> labelValueBeanList = new ArrayList<>();  
  
 if (workflowTask.getTaskType().compareTo(EnumWorkflowTaskType.***预开发票申请***.type) == 0) {  
 labelValueBeanList = workflowInvoiceBeforehandService.getWorkflowDetailsDataByPid(workflowTask.getPid());  
 } else if (workflowTask.getTaskType().compareTo(EnumWorkflowTaskType.***老后台余额转移申请***.type) == 0){  
 labelValueBeanList = workflowOldBackGroundBalanceTransferSercvice.getWorkflowDetailsDataByPid(workflowTask.getPid());  
 }  
  
 //流程明细  
 List<Integer> typeList = new ArrayList<>();  
 typeList.add(EnumWorkflowTaskUserType.***申请人***.type);  
 typeList.add(EnumWorkflowTaskUserType.***审批人***.type);  
 List<WorkflowTaskExamineDetails> apprDataTempList = workflowMapper.findWorkflowTaskExamineDetailsByTaskIdAndAdmTypeList(taskId, typeList);  
 List<WorkflowTaskExamineDetails> apprDataList = new ArrayList<>();  
 WorkflowTaskExamineDetails withdraw = null;  
  
 if (CollectionUtils.isNotEmpty(apprDataTempList)) {  
 Boolean currNodeFlag = false;  
  
 for (WorkflowTaskExamineDetails examineDetails : apprDataTempList) {  
  
  
 if (workflowTask.getCuserType().compareTo(EnumWorkflowTaskCuserType.***广告主***.type) == 0 && examineDetails.getAdmType().compareTo(EnumWorkflowTaskUserType.***申请人***.type) == 0) {  
 examineDetails.setCurrAdmUserFlag(false);  
 String userName = userService.getAuthNameByUserId(workflowTask.getCuserId(),EnumYesNo.***YES***.status);  
 examineDetails.setAdmName("["+workflowTask.getCuserId()+"]" + userName);  
 } else {  
 if (examineDetails.getAdmId().compareTo(admId) == 0) {  
 examineDetails.setCurrAdmUserFlag(true);  
 }  
 }  
 if (!currNodeFlag && examineDetails.getAdmType().compareTo(EnumWorkflowTaskUserType.***审批人***.type) == 0 && examineDetails.getExamineStatus().compareTo(EnumWorkflowTaskExamineStatus.***等待审批***.type) == 0) {  
 examineDetails.setCurrNodeFlag(true);  
 currNodeFlag = true;  
 }  
 if (currNodeFlag && !examineDetails.getCurrNodeFlag() && examineDetails.getAdmType().compareTo(EnumWorkflowTaskUserType.***审批人***.type) == 0 && examineDetails.getExamineStatus().compareTo(EnumWorkflowTaskExamineStatus.***等待审批***.type) == 0){  
 examineDetails.setApprDetailsShowFlag(false);  
 }  
 if (examineDetails.getExamineStatus().compareTo(EnumWorkflowTaskExamineStatus.***撤回作废***.type) == 0){  
 withdraw = examineDetails;  
 continue;  
 }  
 if (examineDetails.getExamineStatus().compareTo(EnumWorkflowTaskExamineStatus.***等待审批***.type) == 0  
 && workflowTask.getStatus().compareTo(EnumWorkflowTaskStatus.***撤回作废***.type) == 0){  
 break;  
 }  
 apprDataList.add(examineDetails);  
  
 if (examineDetails.getExamineStatus().compareTo(EnumWorkflowTaskExamineStatus.***审批拒绝***.type) == 0){  
 //有审批人拒绝了 或者撤回了 流程结束 后面的审批人不再显示了  
 break;  
 }  
 }  
 if (workflowTask.getStatus().compareTo(EnumWorkflowTaskStatus.***撤回作废***.type) == 0){  
 if (withdraw == null){  
 withdraw = workflowTaskExamineDetailsDAO.findTopByTaskIdAndAdmTypeAndExamineStatusOrderByExamineOrderAsc(taskId,EnumWorkflowTaskUserType.***申请人***.type,EnumWorkflowTaskExamineStatus.***撤回作废***.type);  
 }  
 if (withdraw != null) {  
 if (StringUtils.isBlank(withdraw.getAdmName())){  
 SysAdminUser adminUser = sysAdminUserDAO.findOne(withdraw.getAdmId());  
 if (adminUser != null){  
 withdraw.setAdmName(adminUser.getName());  
 }  
 }  
 apprDataList.add(withdraw);  
 }  
 }  
 }  
  
 //抄送  
 typeList.clear();  
 typeList.add(EnumWorkflowTaskUserType.***抄送人***.type);  
 List<WorkflowTaskExamineDetails> ccDataList = workflowMapper.findWorkflowTaskExamineDetailsByTaskIdAndAdmTypeList(taskId, typeList);  
  
 detailsData.setLabelValueList(labelValueBeanList);  
 detailsData.setApprDataList(apprDataList);  
 detailsData.setCcDataList(ccDataList);  
  
 return detailsData;  
 } else {  
 throw new AppException("没有查询到您要查看的记录[id:" + taskId + "]");  
 }  
}

# 12、审核通过(每次审核通过后才会向后一个发消息，然后后一个人才能查看，并审核)

@RequestMapping("passTaskById")  
@ResponseBody  
public ResponseBean passTaskById(Long taskId, String remark) {  
 try {  
 SysAdminUser adminUser = AppSessionHelper.getSessionUser();  
 workflowService.passTaskById(taskId, remark, adminUser);  
 return ResponseBean.buildSuccess();  
 } catch (AppException e) {  
 return ResponseBean.buildFailure(e.getCode(), e.getMessage());  
 } catch (Exception e) {  
 logger.error(e.getMessage(), e);  
 return ResponseBean.buildFailure(ErrorCodeEnum.***系统错误***);  
 }  
}

@Override  
 public void passTaskById(Long taskId, String remark, SysAdminUser adminUser) {  
 //审核通过  
 if (taskId == null) {  
 throw new AppException("没有获取到你处理记录的ID，处理失败");  
 }  
 WorkflowTask workflowTask = workflowTaskDAO.findOne(taskId);  
 if (workflowTask == null) {  
 throw new AppException("没有查询到您处理的记录数据[taskId:" + taskId + "]，处理失败");  
 }  
   
 if (workflowTask.getStatus().compareTo(EnumWorkflowTaskStatus.***审批完成\_同意***.type) == 0  
 || workflowTask.getStatus().compareTo(EnumWorkflowTaskStatus.***审批完成\_拒绝***.type) == 0) {  
 throw new AppException("该记录已审批结束，不可重复操作，请刷新页面查看最新状态");  
 }

#按照提供给的审核顺序进行审核  
 List<WorkflowTaskExamineDetails> apprDetailsList =

workflowTaskExamineDetailsDAO.findByTaskIdAndAdmTypeAndExamineStatusOrderByExamineOrderAsc

(taskId, EnumWorkflowTaskUserType.***审批人***.type, EnumWorkflowTaskExamineStatus.***等待审批***.type);  
 #如果不是第一个审核人则不能审核  
 if (CollectionUtils.isEmpty(apprDetailsList) || apprDetailsList.get(0) == null ||

apprDetailsList.get(0).getAdmId().compareTo(adminUser.getId()) != 0) {  
 throw new AppException("该记录已审批结束或当前节点不允许您处理");  
 }  
  
// 通过后的处理  
// 1.审核详情里的状态改为审核通过  
// 2.给下一位审批人发送消息，如果有下一位审批人的话  
// 3.判断审批人是否已经全部处理完，是的话 WorkflowTask 表 中记录 标记为审核通过状态，并且给申请人以及抄送人发送消息  
  
 WorkflowTaskExamineDetails currNode = apprDetailsList.get(0);  
 currNode.setExamineStatus(EnumWorkflowTaskExamineStatus.***审批通过***.type);  
 if (StringUtils.isNotBlank(remark)) {  
 currNode.setRemark(remark);  
 }  
 currNode.setUdate(new Date());

保存更新状态  
 workflowTaskExamineDetailsDAO.save(currNode);  
  
 String content = "";  
  
 if (apprDetailsList.size() > 1){ #第一个人审核之后，这个时候状态就变成审核中  
   
 workflowTask.setStatus(EnumWorkflowTaskStatus.***审批中***.type);  
 } else {  
 //已全部处理完 按分类处理申请记录数据，安装我们自己提交的审核类型开始处理自己的事情  
 if (workflowTask.getTaskType().compareTo(EnumWorkflowTaskType.***预开发票申请***.type) == 0) {  
 verifyInvoicePlanService.pass(adminUser, workflowTask.getPid());  
 else if (workflowTask.getTaskType().compareTo(EnumWorkflowTaskType.***老后台余额转移申请***.type) == 0) {  
 workflowOldBackGroundBalanceTransferSercvice.changeStatus(workflowTask.getPid(),EnumOldBackGroundBalanceTranferStatus.***已通过***.status);  
 }  
 else {  
 throw new AppException("未知的taskType:" + workflowTask.getTaskType() + "[taskId:" + taskId + "]");  
 }  
  
 workflowTask.setStatus(EnumWorkflowTaskStatus.***审批完成\_同意***.type);  
 workflowTask.setUdate(new Date());  
  
 //获取抄送人列表  
 List<WorkflowTaskExamineDetails> ccDetailsList =

workflowTaskExamineDetailsDAO.findByTaskIdAndAdmType(taskId, EnumWorkflowTaskUserType.***抄送人***.type);  
 #有时候审核通过要通知广告主  
 if (CollectionUtils.isNotEmpty(ccDetailsList)) {  
 String cuserName = "";  
 if (workflowTask.getCuserType() != null &&

workflowTask.getCuserType().compareTo(EnumWorkflowTaskCuserType.***广告主***.type) == 0) {  
 cuserName = userService.getAuthNameByUserId(workflowTask.getCuserId(),EnumYesNo.***YES***.status);  
 } else {  
 SysAdminUser cUser = sysAdminUserDAO.findOne(workflowTask.getCuserId());  
 cuserName = cUser.getName();  
 }  
 content = "您好，"+ cuserName +" 在 " +DateHelper.convertDate2String(workflowTask.getCdate(),DateHelper.***YYYY\_MM\_DD***)+

" 提交的["+EnumWorkflowTaskType.getDes(workflowTask.getTaskType())+"]已审批通过，抄送给您，请知晓。";  
 for (WorkflowTaskExamineDetails cc : ccDetailsList) {  
  
 //给抄送人发送消息  
// System.out.println("给抄送人发送消息-通过");  
   
 //给每个抄送人添加一个抄送我的页面的未读记录  
 Long chenk = workflowTaskUnreadBadgeDAO.countByAdmIdAndTaskIdAndPageType(cc.getAdmId(), taskId, EnumWorkflowPageType.***抄送我的***.type);  
 #如果之前抄送人没有记录则，新增一个未读记录

if (chenk == null || chenk == 0) {  
 WorkflowTaskUnreadBadge unreadBadge = new WorkflowTaskUnreadBadge(taskId, cc.getAdmId(), EnumWorkflowPageType.***抄送我的***.type);  
 workflowTaskUnreadBadgeDAO.save(unreadBadge);  
 }  
 }  
 }  
 }  
 workflowTaskDAO.save(workflowTask);  
  
 }

# 13、审核拒绝（同上面的是一模一样的）