**专业方向综合项目 项目详细设计规约**

**张尧 陈东 姜其升**

**TEam 12**

**MEETING！**

基于微信小程序的协同日程安排应用

**目录**

[1． 引言 2](#_Toc44761150)

[1.1. 编写目的 2](#_Toc44761151)

[1.2. 参考资料 2](#_Toc44761152)

[2． 系统软件架构 3](#_Toc44761153)

[3． 算法设计 7](#_Toc44761154)

[3.1. 时间推荐功能的算法 7](#_Toc44761155)

[4. 类设计 9](#_Toc44761156)

[4.1. EventService 9](#_Toc44761157)

[4.2. Calendar 19](#_Toc44761158)

[4.3. Event 21](#_Toc44761159)

[4.4. EventDetail 25](#_Toc44761160)

[4.5. EventAllInfo 29](#_Toc44761161)

[4.6. EventReminder 32](#_Toc44761162)

[4.7. EventRepeatition 34](#_Toc44761163)

[4.8. UserDetail 37](#_Toc44761164)

[4.9. UserDomain 39](#_Toc44761165)

[4.10. UserCalendar 42](#_Toc44761166)

[4.11. CalendarService 45](#_Toc44761167)

[4.12. UserCalendarService 48](#_Toc44761168)

[4.13. UserService 51](#_Toc44761169)

# 引言

## 编写目的

我们本次将要开发的Meeting系统（以下称本系统）就主要为了解决当今小组开会时的痛点：当一个小组中的所有成员需要敲定一个开会时间时，会进行大量费时费力的讨论，尤其是在小组成员数量多于5人时。在聊天软件中进行开会时间讨论的现状是：每个人主动地提出自己的空闲时间，由其他组员进行比对，只要有一个组员时间有冲突，则又要重复这个过程，效率十分低下。而本系统将从本质出发，提供一个日程管理的功能，使小组成员间能够共享显示每个人的日程安排，并根据所有小组成员的空闲时间自动推荐开会时间，实现快速的开会或其他活动时间确定。

本系统由同济大学软件学院2017级专业方向综合项目第12组成员进行分析、设计开发。小组四人将从软件工程过程架构入手，以软件工程思想讨论、规划、建模、开发、部署，完成本系统构建。

为完成本系统建模阶段的详细设计阶段工作，细化并解释每一个设计类，撰写本文档。  本文档的最终审核版本将作为后续的开发以及测试的参照标准。

## 参考资料

[1]Rogers S.pressman 软件工程：实践者的研究方法[M].北京：机械工业出版社，2011.4：106-151

# 系统软件架构

在进行系统概要设计建模的一开始，我们首先要确定系统的总体架构，即系统的体系结构。系统体系结构提供了一种表示关系，能够有助于所有开发者和利益相关者展开交流，同时它也突出了早期的设计决策。总体架构设计在之后的设计、开发等软件工程工作中，都有着深远的影响。体系结构的类型、风格和模式可以应用于其他系统的设计，并且表示了一组抽象，使得软件工程师能以可预见的方式描述体系结构。

1. **系统环境表示**

我们选择面向对象体系结构作为体系结构风格。以此为依据，我们首先根据需求分析和讨论构建了系统结构环境图（ACD），如图2.1.1所示。

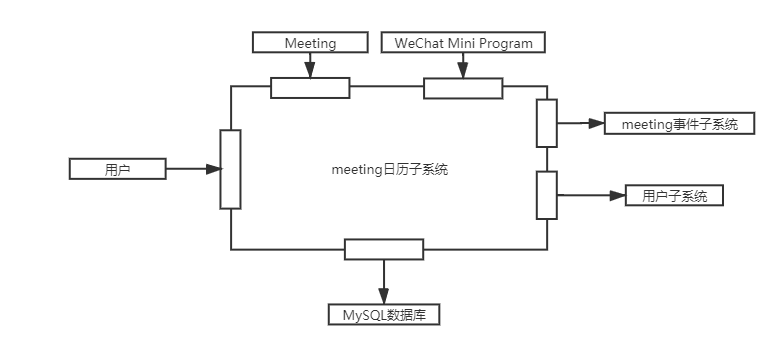


图2.1.1.日历子系统的体系结构环境图

可以从图2.1.1看出，在本系统的日历子系统中，我们设计了它的上级系统、下级系统、统计系统及其参与者。上级系统即为把日历子系统作为某些高层处理方案一部分的系统，如Meeting系统、微信小程序系统等。下级系统即被日历子系统使用，并为完成系统的功能提供必要数据和处理的系统，如MySQL数据库系统。同级系统即在对等的基础上与日历子系统相互作用的系统，如其他的子系统，包括meeting事件子系统和用户子系统。参与者即是与日历子系统产生交互的实体，在本系统中，只有用户是参与者。

其他的事件子系统，包括事件子系统和用户子系统，其体系结构环境图与日历子系统类似，仅有同级系统为其他两个子系统有区别。

1. **定义原型**

原型是表示核心抽象的类或模式，该抽象对于目标系统体系结构设计非常关键。我们通过需求模型的一部分分析类，结合SpringBoot框架的架构，定义了如图2.1.2所示的系统原型架构。

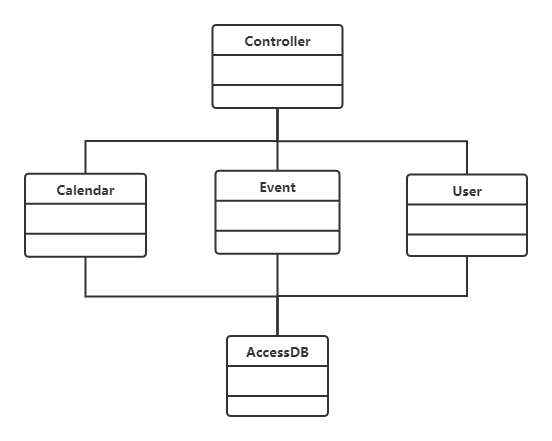


图2.1.2.系统原型的UML关系

如上图所示，我们使用UML符号对每一个原型进行了描述，我们主要将后端系统分为Controller、Service、AccessDB三层架构，对系统体系结构进行初步的抽象构建，在之后，我们会基于原型，进行正式的体系结构构建。

1. **构建系统体系结构**

基于之前完成的需求分析建模、系统体系结构环境图和原型的构建，我们对系统的体系结构有了一个完整的认识，由此我们构建了正式的系统体系结构，在第前几次迭代中，我们已经完成后端的大部分开发，实现了初步的前后端联调。但在设计第三个Sprint的故事功能中，上一次设计的体系结构还不够完善，用于用户鉴权的Redis构件和细化后的UserCalendarService构件被考虑，我们也在系统架构图中添加这些构件，并创建必要的连接。如图2.1.3所示。

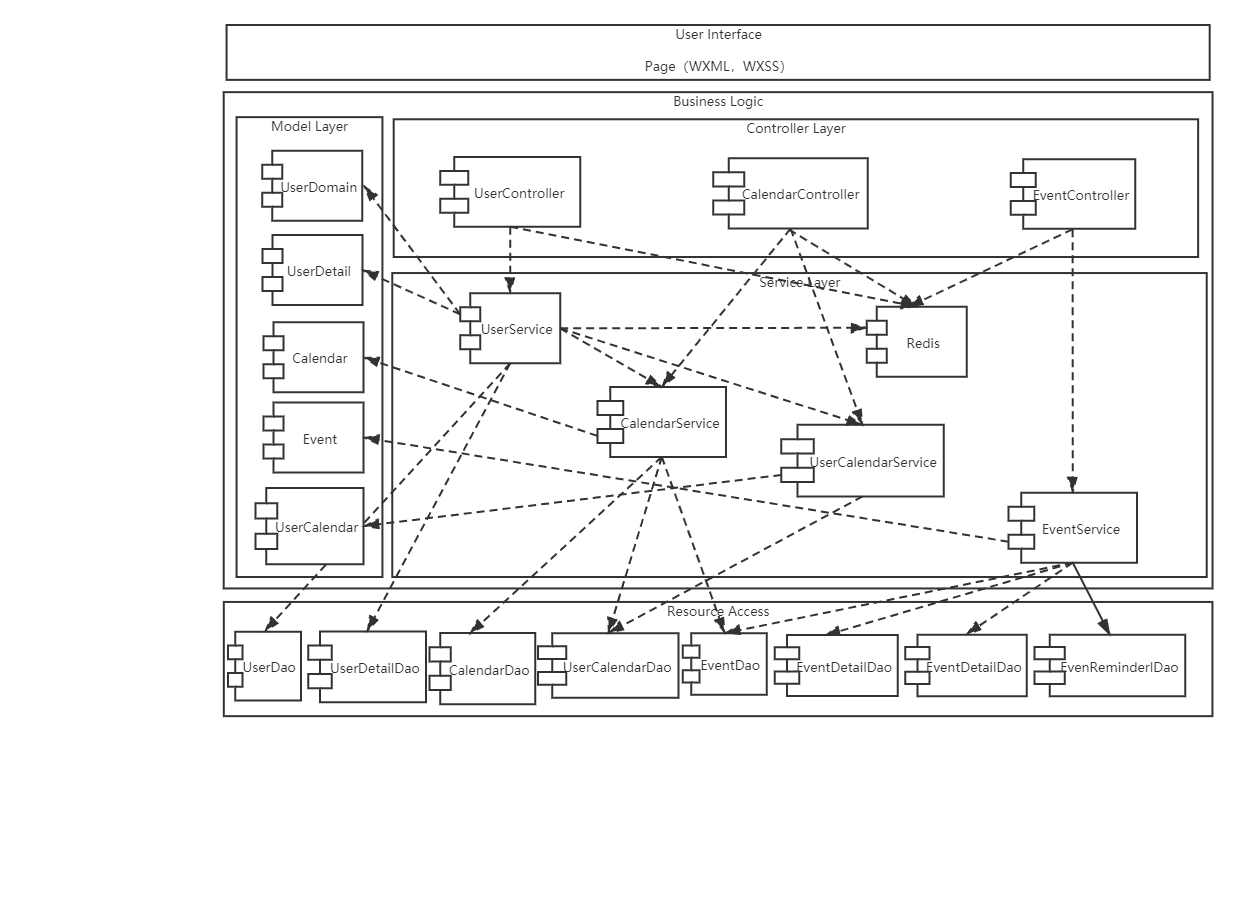


图2.1.3.系统体系结构

本系统体系结构采用经典的三层体系结构，即界面-业务-数据存储的层次。同时，我们在体系结构设计中引入了构件级的设计，按照构件级设计的方法识别设计类并确定协作类集合为一个构件，把构造好的构件图加入系统的体系结构中。

如上图所示，在用户界面层，我们采用基于WXML和WXSS的微信小程序开发技术栈，由于是类HTML5的开发，前端并没有构件产生。后端业务层分离出了Controller层、Service层和Model层，每个层次各司其职。Controller层主要提供前端接入的接口，并整合了Service层提供的一些服务，进行集成。Service层主要提供服务，调用底层Dao，与数据库交互，调用Model层数据，进行基础数据的集成，在本次迭代中主要加入了Redis构件和UserCalendarService构件，分别用于与用户的鉴权和UserCalendar的创建。Model层主要是基础实体类的集合，提供基础数据的定义和基本操作。除此之外，我们把Dao层进行细化，在原有六个构件的基础上，添加了UserCalendarDao、EventDetailedao和EventReminderDao。，它们分别处理不同数据库的数据的增删改查操作。Service层构件通过调用底层的Dao层构件，与数据库进行交互，进行数据的持久化存储。

将构件变成详细的设计类图表示，优化后的整体系统结构如图2.1.4所示。

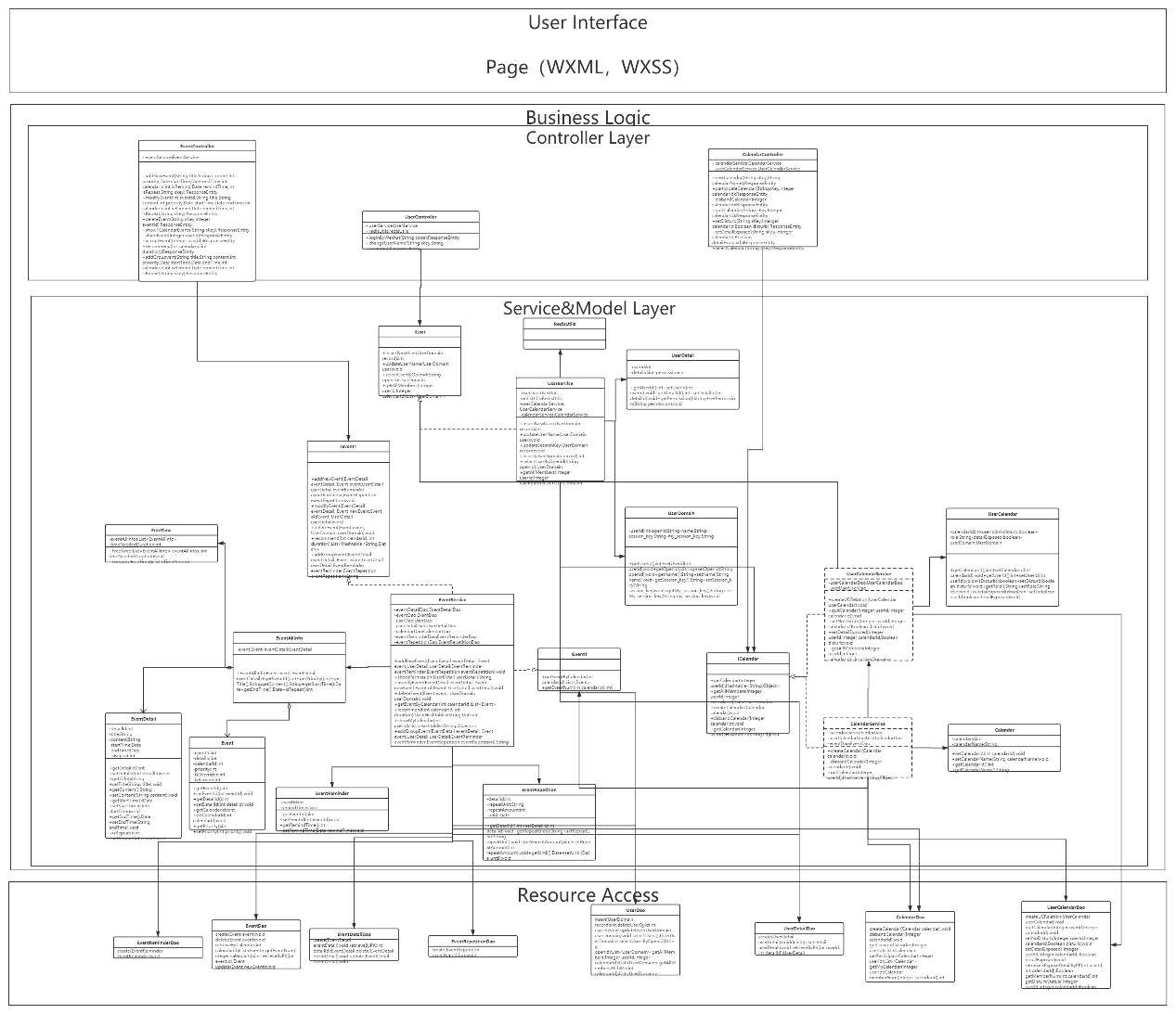


图2.1.4.系统详细体系结构

# 算法设计

由于本项目不涉及高深的算法，选择了一个逻辑比较复杂的进行分析。

## 3.1. 时间推荐功能的算法

**输入：**

某工作组的标识符ID和需要的空闲时间的长度。

**输出：**

空闲时间。

**步骤：**

1、调用DAO获取该工作组的所有成员列表。

2、对每个成员：调用其他方法getAllEventsOfOneUserAfterNow来获得所有用户的所有事件的列表。

3、用2中的所有事件列表和需要开会的时长，构造FreeTime对象。

4、在FreeTime对象中计算空闲时间：

5、为每个事件构造TimePeriod对象，表示某事件发生的时间段。

6、根据事件的开始时间的先后，将一些列时间段放入List中排序。

7、令et=系统当前时间

8、对于每个排序完成的时间段：循环

9、令st=当前处理的时间段的开始时间

10、如果st与et不是同一天：那么et到st之间的这段时间加入空闲时间段列表

11、如果st在et之后，那么et到st这段时间加入空闲时间列表

12、如果et在当前处理的事件时间段的结束时间之后，et更新为当前处理的事件时间段的结束时间

13、直到所有事件时间段都处理完成，结束循环

14、将et到7天后的那一天的最后一时刻这个时间段也加入空闲时间列表。

**流程图**

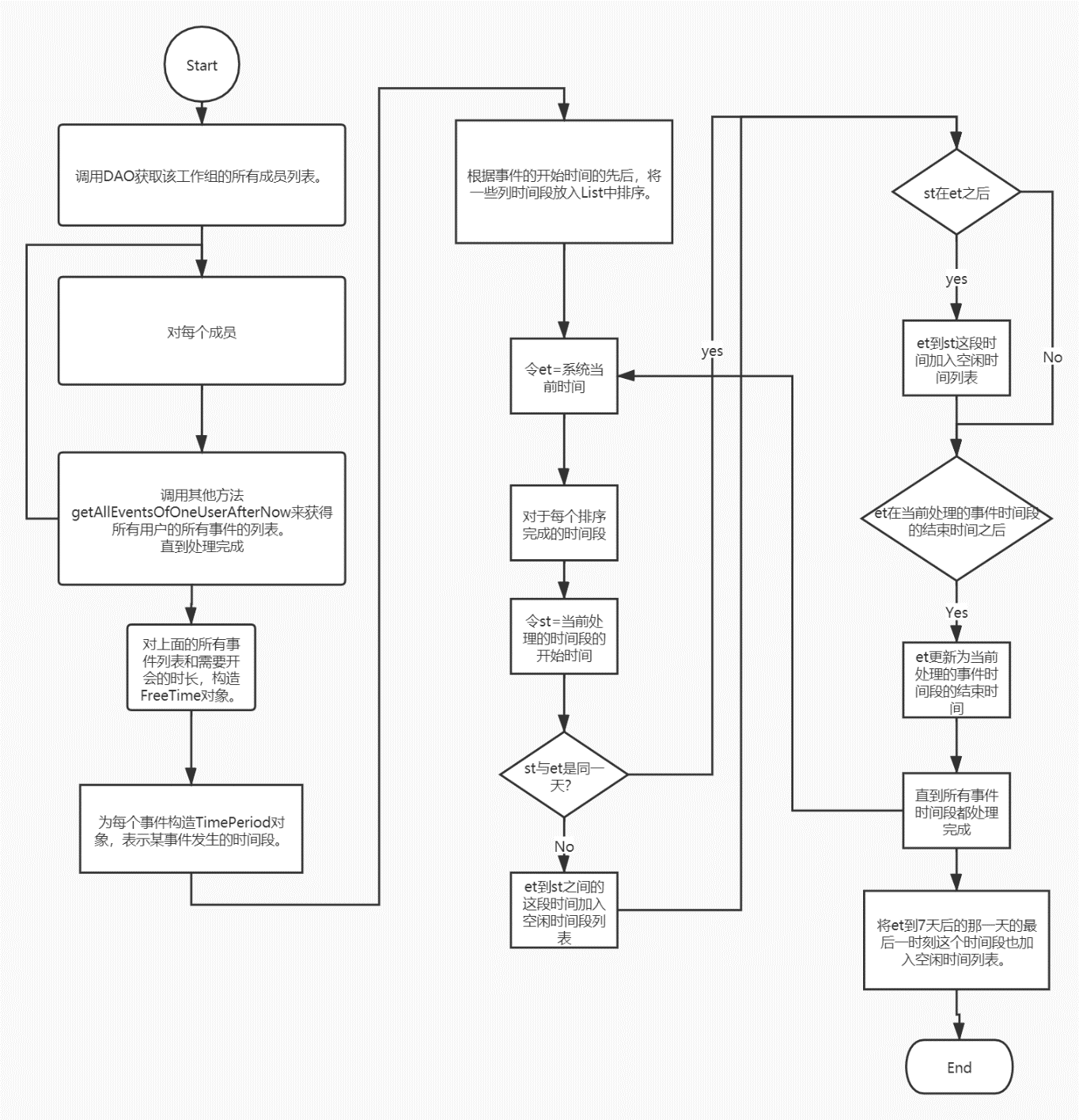


图3.1.算法流程图

# 4. 类设计

构建的项目设计类图如图4.1所示。

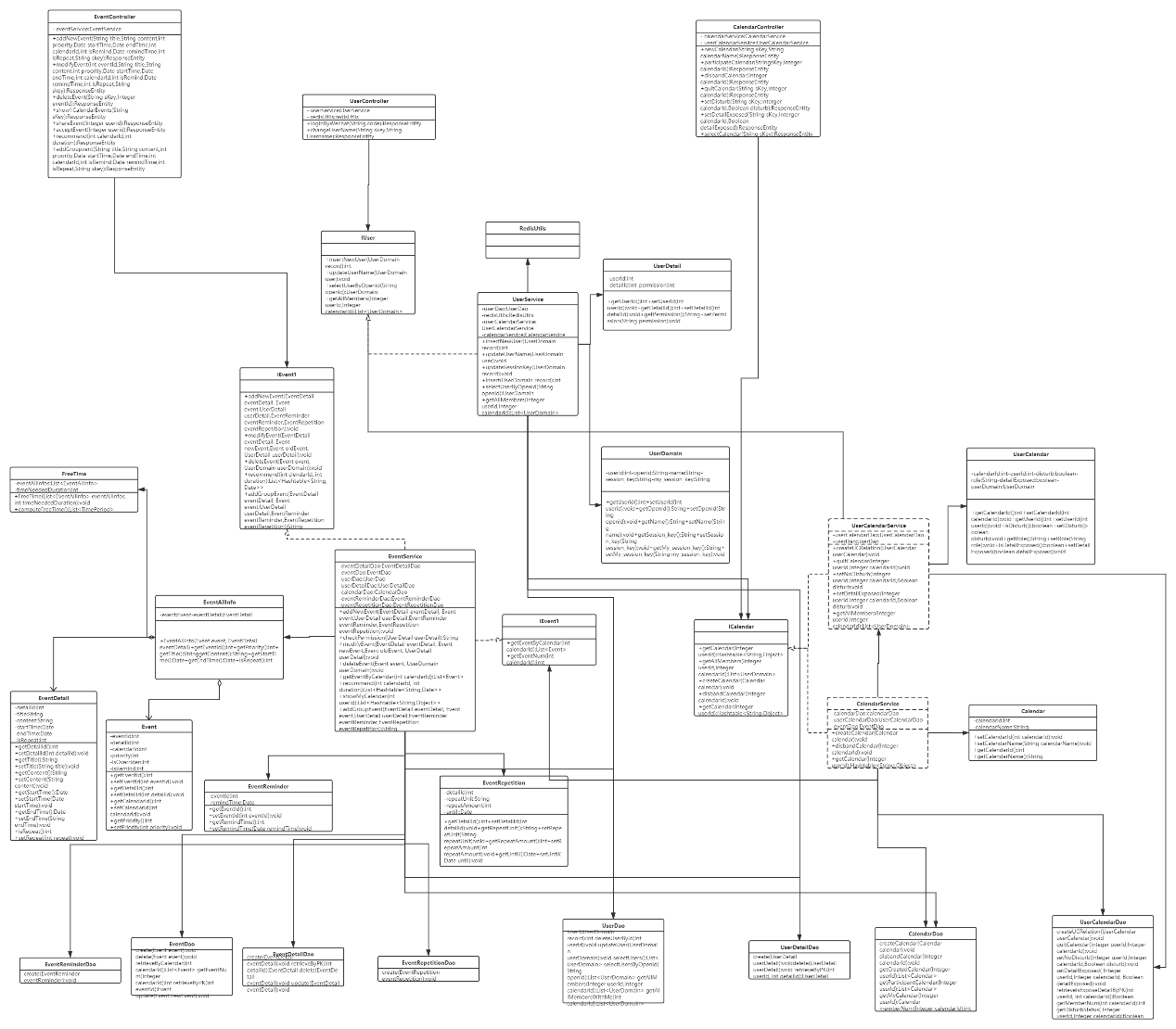


图4.1.系统设计类图

## 4.1. EventService

**程序描述**

com.tongji.meeting.service

public class EventService

有关事件的业务操作。

**功能**

包括添加事件、修改事件、获得某人某日历事件、获得空闲时间、获得用户对事件的权限信息等等操作。

**域**

|  |  |  |
| --- | --- | --- |
| 修饰符和类型 | 名称 | 描述 |
| private int | calendarId; | 日历的标识符ID |
| private int | userId; | 用户的ID |
| private boolean | disturb; | 是否开启勿扰模式 |
| private String | role; | 用户在日历中的角色 |
| private boolean | detailExposed; | 是否开启隐私模式 |

**方法：**

1. addNewEvent

public void addNewEvent(EventDetail eventDetail, Event event,  
 UserDetail userDetail, EventReminder eventReminder,  
 EventRepetition eventRepetition)

添加新事件

**returns**

void

**parameters**

EventDetail eventDetail, Event event, UserDetail userDetail, EventReminder eventReminder, EventRepetition eventRepetition

**实现流程**

1.调用DAO层创建一个event\_detail行

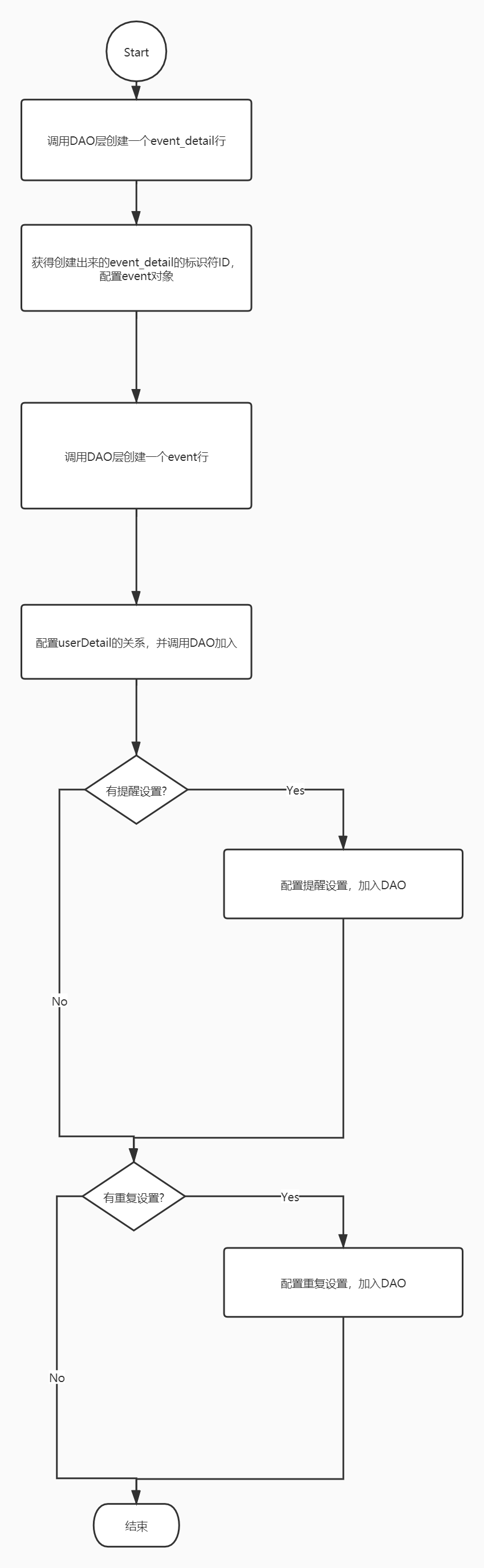
2.获得创建出来的event\_detail的标识符ID，配置event对象

3.调用DAO层创建一个event行

4.配置userDetail的关系，并调用DAO加入

5.如果有提醒设置，配置提醒设置，加入DAO

6.如果，配置提醒设置，加入DAO



1. checkPermission

public String checkPermission(UserDetail userDetail)

获得权限信息

**returns**

String

**parameters**

UserDetail userDetail

**实现流程**

userDetail = userDetailedao.retrieveByPK(userDetail.getUserId(),userDetail.getDetailed());  
return userDetail.getPermission();

1. modifyEvent

public void modifyEvent(EventDetail eventDetail, Event newEvent…)

获得权限信息

**returns**

void

**parameters**

EventDetail eventDetail, Event newEvent, Event oldEvent, UserDetail userDetail

**实现流程**

1.获取修改前的event对象

2.userDetail修改为1中对象的detailed

3.eventDetail设置1中对象的detailed

4.如果用户对于事件的权限是owner，则可以进行更新操作，否则跳至6

5.调用DAO调整event，eventDetail

6.结束

1. deleteEvent

public void deleteEvent(EventDetail eventDetail, UserDomain userDomain

)

删除事件

**returns**

void

**parameters**

EventDetail eventDetail, UserDomain userDomain

**实现流程**

1.获取修改前的event对象

2.userDetail修改为1中对象的detailed

3.eventDetail设置1中对象的detailed

4.如果用户对于事件的权限是owner，则可以进行更新操作，否则跳至6

5.调用DAO删除event，eventDetail

6.结束

1. getEventByCalendar

public List<EventAllInfo> getEventByCalendar (int calendarId)

获取某日历的所有事件

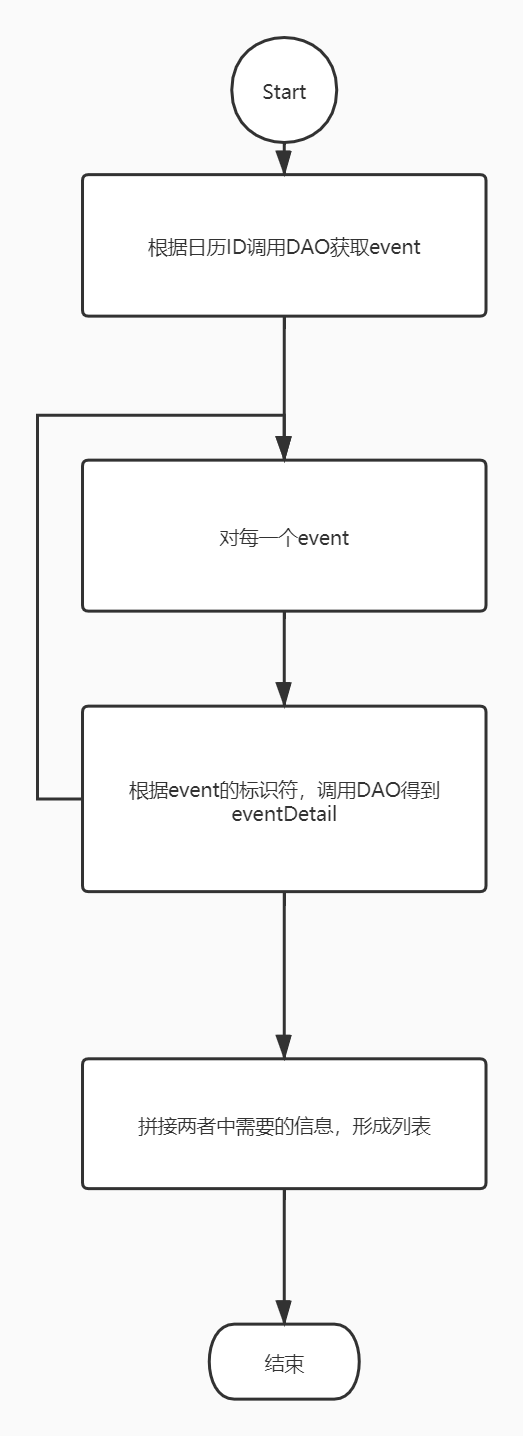
**returns**

List<EventAllInfo>

**parameters**

int calendarId

**实现流程**



1. getEventByCalendarAfterNow

public List<EventAllInfo> getEventByCalendarAfterNow (int calendarId){

获取某日历现在以后的事件

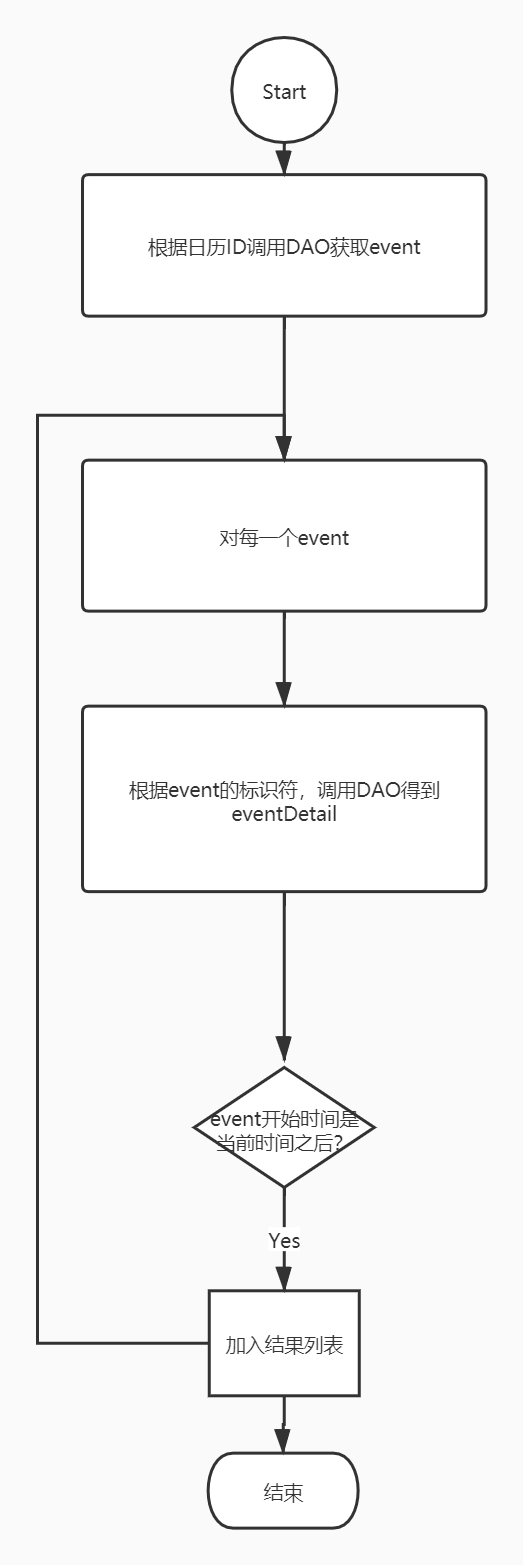
**returns**

List<EventAllInfo>

**parameters**

int calendarId

**实现流程**



1. getAllEventsOfOneUser

public List<EventAllInfo> getAllEventsOfOneUser (UserDomain userDomain)

获取某个用户的所有事件

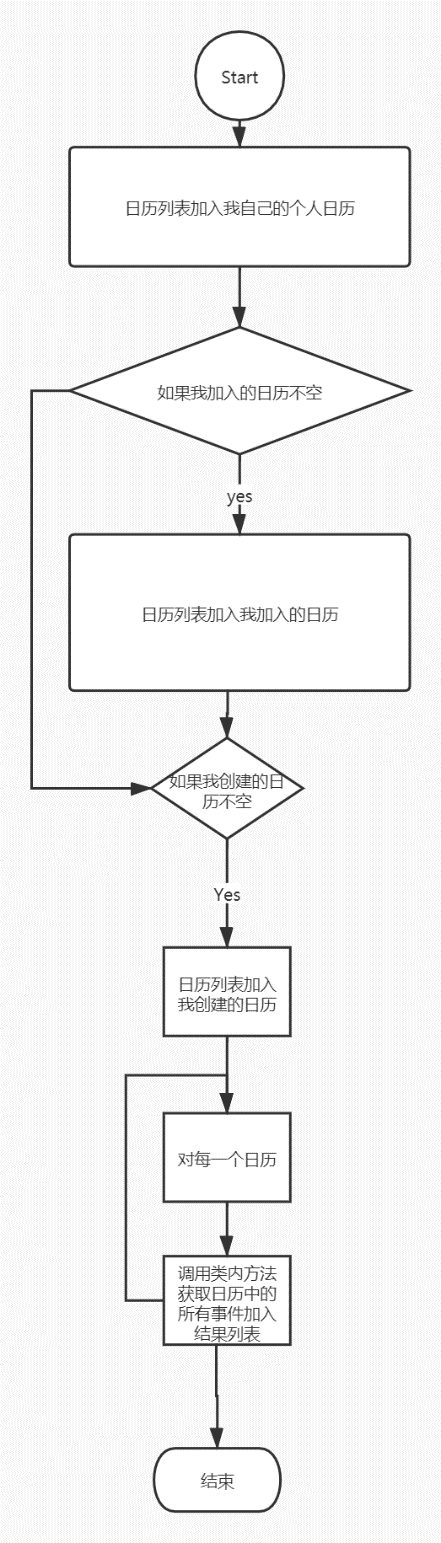
**returns**

List<EventAllInfo>

**parameters**

UserDomain userDomain)

**实现流程**



1. getAllEventsOfOneUserAfterNow

public List<EventAllInfo> getAllEventsOfOneUserAfterNow(UserDomain userDomain)

获取某个用户的所有事件，但要在当前时间之后

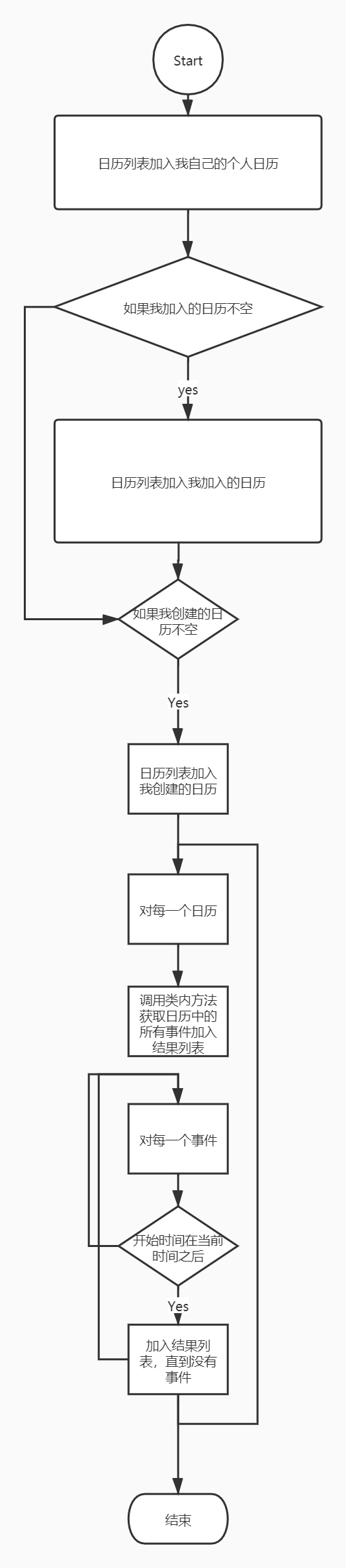
**returns**

List<EventAllInfo>

**parameters**

UserDomain userDomain)

**实现流程**



1. addGroupEvent

public String addGroupEvent(EventDetail eventDetail, Event event, UserDetail userDetail, EventReminder eventReminder, EventRepetition eventRepetition

)

添加组事件

**returns**

String

**parameters**

EventDetail eventDetail, Event event, UserDetail userDetail, EventReminder eventReminder, EventRepetition eventRepetition

**实现流程**

1.在userDetail中获得userId

1.在calendar中获得calendarId

3.调用DAO获取用户的日历中所有事件

4.用获得的事件的开始结束时间构造一个TimePeriod对象

5.对于每一个成员，调用类内**方法**getAllEventsOfOneUser(member);

6.遍历这些event，若和TimePeriod有时间交集，则失败。

7.否则，调用DAO添加event, eventDetail, 提醒或重复设置如果有也添加

8.为组员关联新事件userDetail关系。

1. recommend

public List<Hashtable<String,Date>> recommend(int calendarId, int duration)

推荐组内时间

**returns**

List<Hashtable<String,Date>>

**parameters**

int calendarId, int duration

**实现流程**

1、调用DAO获取该工作组的所有成员列表。

2、对每个成员：调用其他**方法**getAllEventsOfOneUserAfterNow来获得所有用户的所有事件的列表。

3、用2中的所有事件列表和需要开会的时长，构造FreeTime对象。

4、在FreeTime对象中计算空闲时间：

5、为每个事件构造TimePeriod对象，表示某事件发生的时间段。

6、根据事件的开始时间的先后，将一些列时间段放入List中排序。

7、令et=系统当前时间

8、对于每个排序完成的时间段：循环

9、令st=当前处理的时间段的开始时间

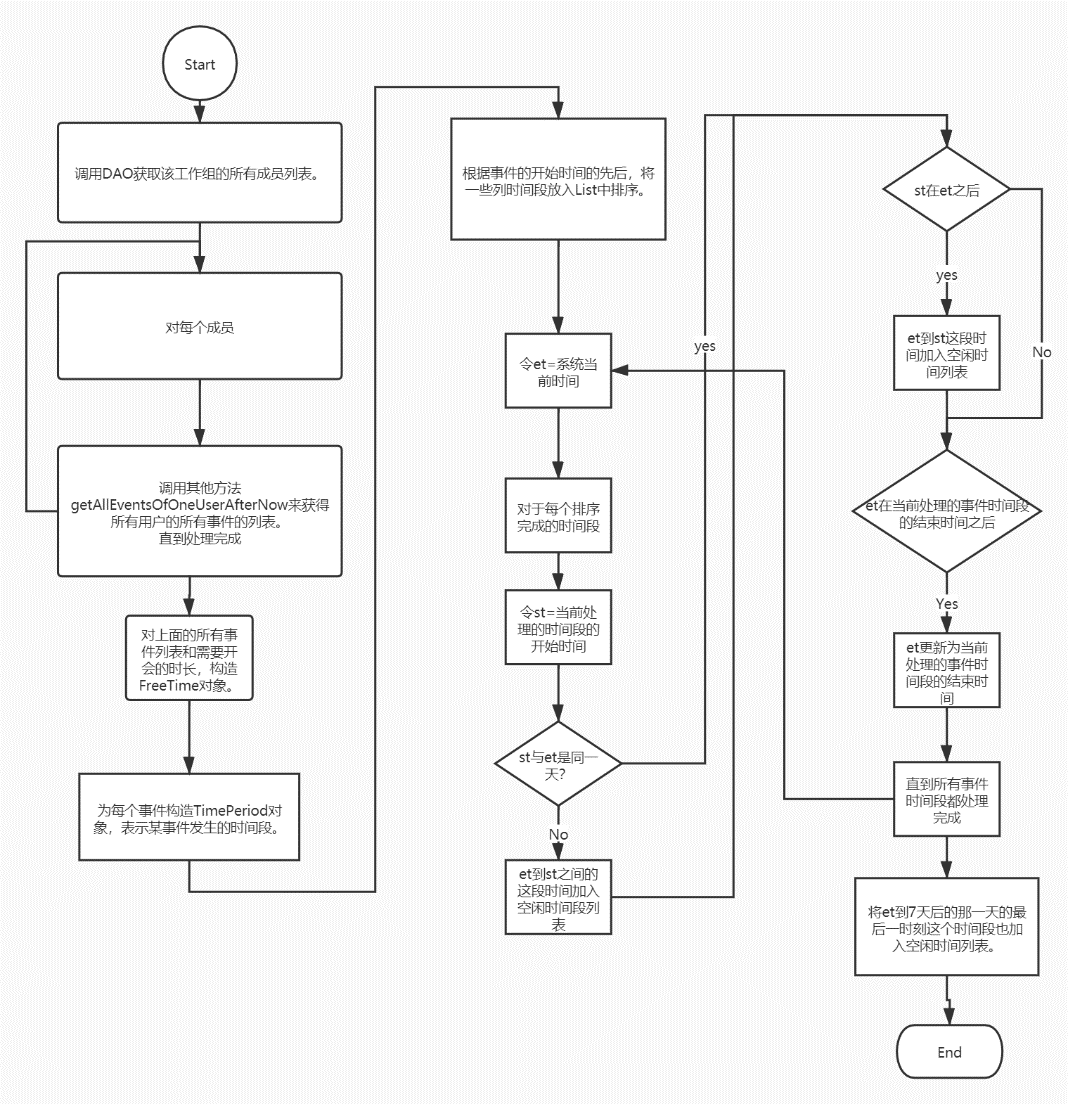
10、如果st与et不是同一天：那么et到st之间的这段时间加入空闲时间段列表

11、如果st在et之后，那么et到st这段时间加入空闲时间列表

12、如果et在当前处理的事件时间段的结束时间之后，et更新为当前处理的事件时间段的结束时间

13、直到所有事件时间段都处理完成，结束循环

14、将et到7天后的那一天的最后一时刻这个时间段也加入空闲时间列表。



## 4.2. Calendar

**程序描述**

com.tongji.meeting.model

public class Calendar

日历

**功能**

对calendarID、calendarName的获取和设置，封装成一个日历对象。

**域**

|  |  |  |
| --- | --- | --- |
| 修饰符和类型 | 名称 | 描述 |
| private int | calendarId | 日历的标识符ID |
| private String | calendarName | 日历的名称 |

**方法**

1. setCalendarId

public void setCalendarId

设置日历的标识符ID。

**returns**

void

**parameters**

int calendarId

**实现流程**

将类内成员变量修改为calendarId = calendarId;

1. setCalendarName

public void setCalendarName

设置日历的名称

**returns**

void

**parameters**

String calendarName

**实现流程**

将类内成员变量修改为calendarName = calendarName;

1. getCalendarId

public int getCalendarId()

获取日历的标识符ID。

**returns**

int

**parameters**

void

**实现流程**

return calendarId;

1. getCalendarName

public String getCalendarName()

获取日历的名字。

**returns**

String

**parameters**

void

**实现流程**

return calendarName;

## 4.3. Event

**程序描述**

com.tongji.meeting.model

public class Event

事件实体类

**功能**

对eventId、detailed、calendarId、priority、isOverriden、isRemind的获取和设置，封装成一个事件对象。

**域**

|  |  |  |
| --- | --- | --- |
| 修饰符和类型 | 名称 | 描述 |
| private int | eventId; | 事件的标识符ID |
| private int | detailed; | 事件详情的标识符ID |
| private int | calendarId; | 日历的标识符ID |
| private int | priority; | 事件优先级 |
| private int | isOverriden; | 是否被覆盖 |
| private int | isRemind; | 是否设置了提醒 |

**方法**

1. getEventId

public int getEventId()

获取事件的标识符ID。

**returns**

int

**parameters**

void

**实现流程**

return eventId;

1. setEventId

public void setEventId(int eventId)

获取事件的标识符ID。

**returns**

void

**parameters**

int eventid

**实现流程**

将类内成员变量修改为eventId = eventId;

1. getDetailIed

public int getDetailIed()

获取事件详情的标识符ID。

**returns**

int

**parameters**

void

**实现流程**

return detailIed;

1. setDetailIed

public int setDetailIed()

设置事件详情的标识符ID。

**returns**

void

**parameters**

int detailed

**实现流程**

将类内成员变量修改为detailed = detailed;

1. getCalendarId

public int getCalendarId()

获取日历的标识符ID。

**returns**

int

**parameters**

void

**实现流程**

return calendarId;

1. setCalendarId

public void setCalendarId (int calendarId)

设置日历的标识符ID。

**returns**

void

**parameters**

int calendarId

**实现流程**

将类内成员变量修改为calendarId = calendarId;

1. getPriority

public int getPriority ()

获取优先级。

**returns**

int

**parameters**

void

**实现流程**

return priority;

1. setPriority

public void setPriority (int priority)

设置优先级。

**returns**

void

**parameters**

int priority

**实现流程**

将类内成员变量修改为priority = priority;

1. getIsOverriden

public int getIsOverriden ()

获取日历的标识符ID。

**returns**

int

**parameters**

void

**实现流程**

return isOverriden;

1. setIsOverriden

public void setIsOverriden (int isOverriden)

设置是否被覆盖。

**returns**

void

**parameters**

int isOverriden

**实现流程**

将类内成员变量修改为isOverriden = isOverriden;

1. setIsRemind

public void setIsRemind (int isRemind)

设置是否有提醒设置。

**returns**

void

**parameters**

int isRemind

**实现流程**

将类内成员变量修改为isRemind = isRemind;

## EventDetail

**程序描述**

com.tongji.meeting.model

public class EventDetail

事件详情的实体类。

**功能**

对detailed、title、content、startTime等等详情字段的获取和设置，封装成一个事件详情对象。

**域**

|  |  |  |
| --- | --- | --- |
| 修饰符和类型 | 名称 | 描述 |
| private int | detailed; | 事件详情的标识符ID |
| private String | title; | 事件标题 |
| private String | content; | 事件内容 |
| private Date | startTime; | 事件开始时间 |
| private Date | endTime; | 事件结束时间呢 |
| private int | isRepeat; | 事件是否重复设置标志 |

**方法**

1. getTitle

public String getTitle ()

获取标题。

**returns**

String

**parameters**

void

**实现流程**

return title;

1. setDetailed

public void setDetailed (int detailed)

设置详情的标识符ID。

**returns**

void

**parameters**

int detailed

**实现流程**

将类内成员变量修改为detailed = detailed;

1. getDetailIed

public int getDetailed ()

获取详情的标识符ID。

**returns**

int

**parameters**

void

**实现流程**

return detailed;

1. setTitle

public void setTitle (String title)

设置标题。

**returns**

void

**parameters**

String title

**实现流程**

将类内成员变量修改为title = title;

1. getContent

public String getContent ()

获取内容。

**returns**

String

**parameters**

void

**实现流程**

return content;

1. setContent

public void setContent (String content)

设置内容。

**returns**

void

**parameters**

String content

**实现流程**

将类内成员变量修改为content = content;

1. getStartTime

public Date getStartTime ()

获取开始时间。

**returns**

Date

**parameters**

void

**实现流程**

return startTime;

1. setStartTime

public void setStartTime (Date startTime)

设置开始时间。

**returns**

void

**parameters**

Date startTime)

**实现流程**

将类内成员变量修改为startTime = startTime;

1. getEndTime

public Date getEndTime ()

获取结束时间。

**returns**

Date

**parameters**

void

**实现流程**

return endTime;

1. setEndTime

public void setEndTime (Date endTime)

设置结束时间。

**returns**

void

**parameters**

Date endTime

**实现流程**

将类内成员变量修改为endTime = endTime

1. isRepeat

public int isRepeat ()

获取是否重复设置。

**returns**

int

**parameters**

void

**实现流程**

return isRepeat;

1. setRepeat

public void setRepeat (int repeat)

获取是否重复设置。

**returns**

void

**parameters**

int repeat

**实现流程**

isRepeat = repeat;

## EventAllInfo

**程序描述**

com.tongji.meeting.model

public class EventAllInfo

融合事件和事件详情的一个类，使得获取事件完整信息更加方便。

**功能**

拼接对应的事件和事件详情，对eventId、detailed、calendarId、priority、isOverriden、isRemind的获取和设置，封装成一个完整的对象。

**域**

|  |  |  |
| --- | --- | --- |
| 修饰符和类型 | 名称 | 描述 |
| private Event | event | Event类的一个对象 |
| private EventDetail | eventDetail | EventDetail类的一个对象 |

**方法**

1. 构造方法EventAllInfo

public EventAllInfo(Event event, EventDetail eventDetail)

用一个Event对象和EventDetail对象构造一个EventAllInfo。

**returns**

**parameters**

Event event, EventDetail eventDetail

**实现流程**

将类内成员变量修改为event = event;  
将类内成员变量修改为eventDetail = eventDetail;

1. getEventId

public int getEventId ( )

获取事件的标识符ID。

**returns**

int

**parameters**

void

**实现流程**

return event.getEventId();

1. getPriority

public int getPriority ( )

获取事件的优先级。

**returns**

int

**parameters**

void

**实现流程**

return event.getPriority();

1. getTitle

public String getTitle ( )

获取事件的标题。

**returns**

String

**parameters**

void

**实现流程**

return eventDetail.getTitle();

1. getContent

public String getContent ( )

获取事件的内容。

**returns**

String

**parameters**

void

**实现流程**

return eventDetail.getContent();

1. getStartTime

public Date getStartTime ( )

获取事件的开始时间。

**returns**

Date

**parameters**

void

**实现流程**

return eventDetail.getStartTime();

1. getEndTime

public Date getEndTime ( )

获取事件的结束时间。

**returns**

Date

**parameters**

void

**实现流程**

return eventDetail.getEndTime();

isRepeat

public int isRepeat ()

获取事件是否重复设置。

**returns**

int

**parameters**

void

**实现流程**

return eventDetail.isRepeat();

## EventReminder

**程序描述**

com.tongji.meeting.model

public class EventReminder

事件提醒设置的实体类。

**功能**

获取和设置事件对应的提醒设置。

**域**

|  |  |  |
| --- | --- | --- |
| 修饰符和类型 | 名称 | 描述 |
| private int | eventId | 事件的标识符ID |
| private Date | remindTime | 提醒时间 |

**方法**

1. getEventId

public int getEventId ( )

获取事件的标识符ID。

**returns**

int

**parameters**

void

**实现流程**

return event.getEventId();

1. setEventId

public void setEventId(int eventId)

获取事件的标识符ID。

**returns**

void

**parameters**

int eventid

**实现流程**

将类内成员变量修改为eventId = eventId;

1. getRemindTime

public Date getRemindTime()

获取事件提醒的时间。

**returns**

Date

**parameters**

void

**实现流程**

return remindTime;

1. setRemindTime

public void setRemindTime(Date remindTime)

设置事件提醒的时间。

**returns**

void

**parameters**

Date remindTime

**实现流程**

将类内成员变量修改为remindTime = remindTime;

## EventRepeatition

**程序描述**

com.tongji.meeting.model

public class EventRepetition

事件重复设置的实体类。

**功能**

获取和设置事件对应的重复设置。

**域**

|  |  |  |
| --- | --- | --- |
| 修饰符和类型 | 名称 | 描述 |
| private int | detailed; | 事件详情的标识符ID |
| private String | repeatUnit; | 重复单位 |
| private int | repeatAmount; | 重复数量 |
| private Date | until; | 停止日期 |

**方法**

1. getDetailed

public int getDetailed ()

获取详情的标识符ID。

**returns**

int

**parameters**

void

**实现流程**

return detailed;

1. setDetailed

public void setDetailed (int detailed)

设置详情的标识符ID。

**returns**

void

**parameters**

int detailed

**实现流程**

将类内成员变量修改为detailed = detailed;

1. getRepeatUnit

public int getRepeatUnit ()

获取重复单位。

**returns**

String

**parameters**

void

**实现流程**

return RepeatUnit;

1. setRepeatUnit

public void setRepeatUnit (int RepeatUnit)

设置重复单位。

**returns**

void

**parameters**

String RepeatUnit

**实现流程**

将类内成员变量修改为RepeatUnit = RepeatUnit;

1. getRepeatAmount

public int getRepeatAmount ()

获取重复数量。

**returns**

int

**parameters**

void

**实现流程**

return RepeatAmount;

1. setRepeatAmount

public void setRepeatAmount (int RepeatAmount)

设置重复数量。

**returns**

void

**parameters**

int RepeatAmount

**实现流程**

将类内成员变量修改为RepeatAmount = RepeatAmount;

1. getUntil

public int getUntil ()

获取停止重复日期。

**returns**

int

**parameters**

void

**实现流程**

return until;

1. setUntil

public void setUntil (int until)

设置停止重复日期。

**returns**

void

**parameters**

int until

**实现流程**

将类内成员变量修改为until = until;

## UserDetail

**程序描述**

com.tongji.meeting.model

public class UserDetail

用户对应详情关系的实体类。

**功能**

获取和设置用户对应详情的关系及权限。

**域**

|  |  |  |
| --- | --- | --- |
| 修饰符和类型 | 名称 | 描述 |
| private int | userId; | 用户的标识符ID |
| private int | detailed; | 详情的标识符ID |
| private String | permission; | 权限 |

**方法**

1. getDetailed

public int getDetailed ()

获取详情的标识符ID。

**returns**

int

**parameters**

void

**实现流程**

return detailed;

1. setDetailed

public void setDetailed (int detailed)

设置详情的标识符ID。

**returns**

void

**parameters**

int detailed

**实现流程**

将类内成员变量修改为detailed = detailed;

1. getPermission

public String get Permission()

获取权限。

**returns**

String

**parameters**

void

**实现流程**

return permission

1. setPermission

public void setPermission

设置权限。

**returns**

void

**parameters**

String permission

**实现流程**

将类内成员变量修改为permission = permission;

1. getUserId

public int getUserId ()

获取详情的标识符ID。

**returns**

int

**parameters**

void

**实现流程**

return userId;

1. setUserId

public void setUserId (int userId)

设置详情的标识符ID。

**returns**

void

**parameters**

int userId

**实现流程**

将类内成员变量修改为userId = userId;

## UserDomain

**程序描述**

com.tongji.meeting.model

public class UserDomain

用户实体类。

**功能**

获取和设置用户信息，登录处理信息。

**域**

|  |  |  |
| --- | --- | --- |
| 修饰符和类型 | 名称 | 描述 |
| private int | userid; | 用户的标识符ID |
| private String | openid; | 微信对应用户的ID |
| private String | name; | 用户名 |
| private String | session\_key; |  |
| private String | my\_session\_key; |  |

**方法**

1. getUserId

public int getUserId ()

获取详情的标识符ID。

**returns**

int

**parameters**

void

**实现流程**

return userId;

1. setUserId

public void setUserId (int userId)

设置详情的标识符ID。

**returns**

void

**parameters**

int userId

**实现流程**

将类内成员变量修改为userId = userId;

1. getOpenId

public String getOpenId()

获取微信OpenId。

**returns**

String

**parameters**

void

**实现流程**

return openId

1. setOpenId

public void setOpenId

设置微信OpenId。

**returns**

void

**parameters**

String openId

1. getName

public String getName()

获取微信Name。

**returns**

String

**parameters**

void

**实现流程**

return name

1. setName

public void setName

设置微信Name。

**returns**

void

**parameters**

String name

1. getSessionKey

public String getSessionKey()

获取微信SessionKey。

**returns**

String

**parameters**

void

**实现流程**

return sessionKey

1. setSessionKey

public void setSessionKey

设置微信SessionKey。

。

**returns**

void

**parameters**

String sessionKey

## UserCalendar

**程序描述**

com.tongji.meeting.model

public class UserCalendar

用户和日历关系、特征关系的实体类。

**功能**

获取和设置用户和日历的关系。

**域**

|  |  |  |
| --- | --- | --- |
| 修饰符和类型 | 名称 | 描述 |
| private int | calendarId; | 日历的标识符ID |
| private int | userId; | 用户的ID |
| private boolean | disturb; | 是否开启勿扰模式 |
| private String | role; | 用户在日历中的角色 |
| private boolean | detailExposed; | 是否开启隐私模式 |

**方法**

1. getCalendarId

public int getCalendarId()

获取日历的标识符ID。

**returns**

int

**parameters**

void

**实现流程**

return calendarId;

1. setCalendarId

public void setCalendarId

设置日历的标识符ID。

**returns**

void

**parameters**

int calendarId

**实现流程**

将类内成员变量修改为calendarId = calendarId;

1. getUserId

public int getUserId ()

获取详情的标识符ID。

**returns**

int

**parameters**

void

**实现流程**

return userId;

1. setUserId

public void setUserId (int userId)

设置详情的标识符ID。

**returns**

void

**parameters**

int userId

**实现流程**

将类内成员变量修改为userId = userId;

1. isDisturb

public int isDisturb()

获取是否开启勿扰模式。

**returns**

Boolean

**parameters**

void

**实现流程**

return disturb;

1. setDisturb

public void setDisturb(Boolean disturb)

设置详情的标识符ID。

**returns**

void

**parameters**

Boolean disturb

**实现流程**

将类内成员变量修改为 disturb = disturb;

1. getRole

public String getRole()

获取日历的名字。

**returns**

String

**parameters**

void

**实现流程**

return role;

1. setRole

public void setRole

设置日历的名称。

**returns**

void

**parameters**

String role

**实现流程**

将类内成员变量修改为role = role;

1. getCalendarId

public int getCalendarId()

获取日历的标识符ID。

## CalendarService

**程序描述**

com.tongji.meeting.service

public class CalendarService

有关日历的业务操作。

**功能**

包括添加日历，修改日历，获取某人创建的日历等等操作。

**域**

|  |  |  |
| --- | --- | --- |
| 修饰符和类型 | 名称 | 描述 |
| private int | calendarId; | 日历的标识符ID |
| private int | userId; | 用户的ID |
| private boolean | disturb; | 是否开启勿扰模式 |
| private String | role; | 用户在日历中的角色 |
| private boolean | detailExposed; | 是否开启隐私模式 |

**方法**

1. createCalendar

public void createCalendar(Calendar calendar)

创建一个日历，根据一个日历对象。

**returns**

void

**parameters**

Calendar calendar

**实现流程**

calendarDao.createCalendar(calendar);

1. disbandCalendar

public void disbandCalendar(Integer calendarId)

解散一个日历，根据日历的标识符ID。

**returns**

void

**parameters**

Integer calendarId

**实现流程**

calendarDao.disbandCalendar(calendarId);

1. getCreatedCalendar

public Hashtable<String,Object> getCreatedCalendar(Integer userId)

获取某人创建的日历对象，并转换为JSON格式

**returns**

Hashtable<String,Object>

**parameters**

Integer userId

**实现流程**

List<Calendar> calendarList=calendarDao.getCreatedCalendar(userId);  
List<Integer> memberNum=new ArrayList<>();  
List<Integer> eventNum=new ArrayList<>();  
List<Boolean> disturbModes=new ArrayList<>();  
int calendarId;  
for(Calendar calendar:calendarList){  
 calendarId=calendar.getCalendarId();  
 memberNum.add(userCalendarDao.getMemberNum(calendarId));  
 eventNum.add(eventDao.getEventNum(calendarId));  
 disturbModes.add(userCalendarDao.getDisturbStatus(userId,calendarId));  
}  
Hashtable<String,Object> createdCalendar = new Hashtable<>();  
createdCalendar.put("calendarList",calendarList);  
createdCalendar.put("memberNum",memberNum);  
createdCalendar.put("eventNum",eventNum);  
createdCalendar.put("disturbModes",disturbModes);  
return createdCalendar;

1. getParticipantCalendar

public Hashtable<String,Object> getParticipantCalendar

(Integer userId)

获取某人加入的的日历对象，并转换为JSON需要的格式

**returns**

Hashtable<String,Object>

**parameters**

Integer userId

**实现流程**

List<Calendar> calendarList=calendarDao.getParticipantCalendar(userId);  
List<Integer> memberNum=new ArrayList<>();  
List<Integer> eventNum=new ArrayList<>();  
List<Boolean> disturbModes=new ArrayList<>();  
int calendarId;  
for(Calendar calendar:calendarList){  
 calendarId=calendar.getCalendarId();  
 memberNum.add(userCalendarDao.getMemberNum(calendarId));  
 eventNum.add(eventDao.getEventNum(calendarId));  
 disturbModes.add(userCalendarDao.getDisturbStatus(userId,calendarId));  
}  
Hashtable<String,Object> participantCalendar = new Hashtable<>();  
participantCalendar.put("calendarList",calendarList);  
participantCalendar.put("memberNum",memberNum);  
participantCalendar.put("eventNum",eventNum);  
participantCalendar.put("disturbModes",disturbModes);  
return participantCalendar;

1. getMyCalendar

public Calendar getMyCalendar(Integer userId)

获取我的个人日历对象，并转换为JSON需要的格式

**returns**

Calendar

**parameters**

Integer userId

**实现流程**

return calendarDao.getMyCalendar(userId);

## UserCalendarService

**程序描述**

com.tongji.meeting.service

public class UserCalendarService

实现用户-日历关系维护的类

**功能**

得到用户和日历间的关系（属于关系），新建关系，获取权限信息等等**功能**。

**域**

|  |  |  |
| --- | --- | --- |
| 修饰符和类型 | 名称 | 描述 |
| private int | calendarId; | 日历的标识符ID |
| private int | userId; | 用户的ID |
| private boolean | disturb; | 是否开启勿扰模式 |
| private String | role; | 用户在日历中的角色 |
| private boolean | detailExposed; | 是否开启隐私模式 |

**方法**

1. createUCRelation

public void createUCRelation (UserCalendar userCalendar)

创建用户日历间的关系

**returns**

void

**parameters**

UserCalendar userCalendar

**实现流程**

userCalendarDao.createUCRelation(userCalendar);

1. quitCalendar

public void quitCalendar (Integer userId,Integer calendarId)

退出日历

**returns**

void

**parameters**

Integer userId,Integer calendarId

**实现流程**

userCalendarDao.quitCalendar(userId,calendarId)

1. setNoDisturb

public void setNoDisturb (Integer userId,Integer calendarId)

设置用户对日历的勿扰模式

**returns**

void

**parameters**

Integer userId,Integer calendarId

**实现流程**

userCalendarDao.setNoDisturb(userId,calendarId,disturb);

1. setDetailExposed

public void setDetailExposed(Integer userId,Integer calendarId,Boolean disturb)

设置用户对日历的隐私模式

**returns**

void

**parameters**

Integer userId,Integer calendarId,Boolean disturb

**实现流程**

userCalendarDao.setDetailExposed(userId,calendarId,disturb);

1. getAllMembers

public List<UserDomain> getAllMembers(Integer userId,Integer calendarId)

获取某日历中全体成员

**returns**

List<UserDomain>

**parameters**

Integer userId,Integer calendarId

**实现流程**

return userDao.getAllMembers(userId,calendarId);

1. getDisturbStatus

public boolean getDisturbStatus(Integer userId,Integer calendarId)

获取对日历的勿扰模式信息

**returns**

Boolean

**parameters**

Integer userId,Integer calendarId

**实现流程**

return userCalendarDao.getDisturbStatus(userId,calendarId);

## UserService

**程序描述**

com.tongji.meeting.service

public class UserService

有关用户的业务操作。

**功能**

包括添加用户、有关用户的session\_key等等操作。

**域**

|  |  |  |
| --- | --- | --- |
| 修饰符和类型 | 名称 | 描述 |
| private int | calendarId; | 日历的标识符ID |
| private int | userId; | 用户的ID |
| private boolean | disturb; | 是否开启勿扰模式 |
| private String | role; | 用户在日历中的角色 |
| private boolean | detailExposed; | 是否开启隐私模式 |

**方法**

1. insertNewUser

public int insertNewUser (UserDomain record)

插入新用户

**returns**

int

**parameters**

UserDomain record

**实现流程**

int re = userDao.insert(record);  
updateSessionKey(record);  
int newCalendarId;  
Calendar calendar=new Calendar();  
UserCalendar userCalendar=new UserCalendar();  
calendar.setCalendarName("我的日历");  
userCalendar.setUserId(record.getUserid());  
userCalendar.setDisturb(false);  
userCalendar.setRole("mine");  
userCalendar.setDetailExposed(false);  
calendarService.createCalendar(calendar);  
newCalendarId=calendar.getCalendarId();  
userCalendar.setCalendarId(newCalendarId);  
userCalendarService.createUCRelation(userCalendar);  
return re;

1. updateUserName

public void updateUserName (UserDomain user){

更新用户名字

**returns**

void

**parameters**

UserDomain user

**实现流程**

userDao.updateUser(user);

1. updateSessionKey

public void updateSessionKey (UserDomain record)

更新用户SessionKey

**returns**

void

**parameters**

UserDomain record

**实现流程**

HashMap<String, Object> info = new HashMap<>();  
info.put("userid", record.getUserid());  
info.put("openid", record.getOpenid());  
info.put("session\_key", record.getSession\_key());  
redisUtils.hmset(record.getMy\_session\_key(), info, GlobalValues.mySessionTime);

1. selectUserByOpenid

public UserDomain selectUserByOpenid (String openid)

更新用户名字

**returns**

UserDomain

**parameters**

String openid

**实现流程**

List<UserDomain> records = userDao.selectUsersByOpenid(openid);  
if (records.size() == 0){  
 return null;  
}else {  
 return records.get(0);  
}