软件设计文档模板

目录

[序言 2](#_Toc268020283)

[编写目的 2](#_Toc268020284)

[背景 2](#_Toc268020285)

[文档约定 2](#_Toc268020286)

[可行性分析 3](#_Toc268020287)

[要求目标 3](#_Toc268020288)

[条件限定和假设 3](#_Toc268020289)

[可行性具体分析（包括资源，技术，实现等方面分析） 4](#_Toc268020290)

[设计概述 4](#_Toc268020291)

[系统概述（包括包图） 5](#_Toc268020292)

[用例描述 5](#_Toc268020293)

[用户注册 7](#_Toc268020294)

[用户登陆 7](#_Toc268020295)

[添加任务 7](#_Toc268020296)

[编辑任务 7](#_Toc268020297)

[删除任务 7](#_Toc268020298)

[清空垃圾箱 7](#_Toc268020299)

[添加联系人 7](#_Toc268020300)

[删除联系人 7](#_Toc268020301)

[完成任务 7](#_Toc268020302)

[设计限制（如系统只能采用java语言编写，在eclipse下用等等） 14](#_Toc268020303)

[运行环境 15](#_Toc268020304)

[详细设计类图 15](#_Toc268020305)

[*用户接口设*计（界面设计图例等） 16](#_Toc268020306)

[数据库设计（可以用表或者数据库实体关系图表示） 17](#_Toc268020307)

[类的接口设计（各个类在编写是必须服从那些接口限制） 23](#_Toc268020308)

[参考文档 24](#_Toc268020309)

# 序言

## 编写目的

本文档提供了整个系统的详细设计。具体包括体系架构设计、界面设计，详细设计以及各个模块间的接口约定。其主要用处是为具体编码实现提供一份“蓝图”，同时也为后期维护、扩展提供了参考。

## 背景

本项目名为“WindMill”，由WindMill开发团队负责开发，具体团队成员情况请参考相关项目团队信息。项目主要基于南京大学软件学院暑期项目实训中心。

项目要实现的是一个称作“GTDT”的个人事务管理系统，GTDT具体含义请参考《需求文档》。主要参照谷歌公司下的一项服务“DoIt“的在线任务管理系统，具体请参考以下网址:

<http://i.doit.im>

## 文档约定

GTDT： Get Things Done Tool

MVC： Model View Control

# 可行性分析

## 要求目标

要求做一个“GTDT”的个人事物管理软件，既采用“Get Things Done”的思想来管理个人事物。

模仿 doit 平台的各种功能，实现基础功能的基础上可以进行改善，做到提高用户体验效果，帮助用户管理个人目的的功能。

功能包括：

1. 用户注册登录登出。
2. 完善个人资料。
3. 添加任务，任务属性包括名称，开始时间，截止时间，备注，情景，项目，提醒与否，标签，是否转发，优先级。
4. 对人物进行分类。
5. 搜索任务。
6. 添加联系人。
7. 提供联系人之间任务转发功能。

## 条件限定和假设

1. 同时操作最大任务不超过50个。
2. 用户具有一定的GetThingsDone的思想基础，对任务的归档方法有一定的了解。
3. 每个用户都是在Windows平台下使用该系统。

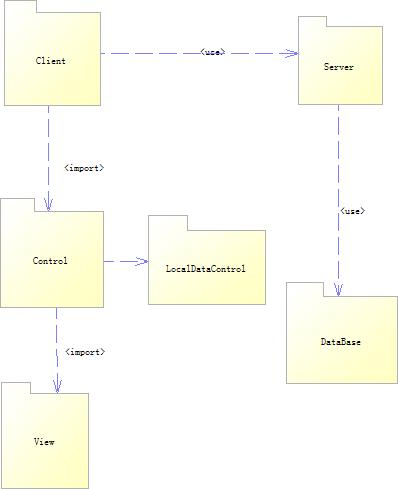
## 可行性具体分析（包括资源，技术，实现等方面分析）

1. 在资源方面，本项目采用PSP方法，在项目开始阶段，通过评估手段，计算本项目需要时间，对项目进行了严密的安排计划，同时根据编程人员的有效工作时间安排，完全能够满足本项目需求
2. 在技术方面，本项目采用所有工作人员都熟练掌握的高级语言java和标准SQL语言，大家不需要重新学习新的技术，就可以完全完成系统实现。
3. 在实现方面，采用MVC架构模式，在网络层采用java语言中Socket和SeverSocket来实现信息交互，信息存储采用标准SQL语言来与MySQL数据库平台交互，界面采用java中Swing各种组件，能够很好的实现该系统的各种功能。

## 设计概述

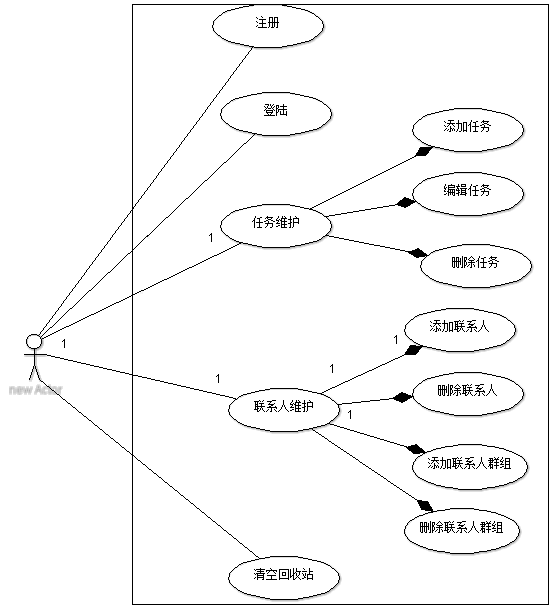
主体采用MVC模式，C/S作为网络连接模式。整体架构分为6个包，分别为Client(客户端登陆) ，View（GUI客户端操作），Control(控制客户端与server端交互)， LocalDataControl(控制本地数据存储和与服务器同步)，Server(服务器端)，DataBase(与服务器交互，封装数据库操作)。

## 系统概述（包括包图）



## 用例描述

用例图如下所示：

**关键任务：**

用户注册

用户登陆

添加任务

编辑任务

删除任务

清空垃圾箱

添加联系人

删除联系人

完成任务

书写以下用例的描述:

|  |  |
| --- | --- |
| 用例名称 | 注册用户 |
| 用例编号 | H001 |
| 执行者 | 游客 |
| 前置条件 | 游客登陆注册界面 |
| 后置条件 | 游客注册为用户，登录网站 |
| 涉众利益 | 每个游客都可以简单的进行操作 |
| 基本路径 | 1. 显示注册需要的信息。 2. 用户填入注册需要的信息。 3. 系统检测用户名信息 4. 注册信息成功，进入登陆后的界面 |
| 扩展 | 1. 用户输入的信息有错误时，系统提示信息错误。 2. 返回 |
| 字段列表 | 用户名，密码。 |
| 设计规则 | 用户名唯一。 |
| 未解决问题 |  |

|  |  |
| --- | --- |
| 用例名称 | 用户登录 |
| 用例编号 | H002 |
| 执行者 | 游客 |
| 前置条件 | 游客进入登陆界面。 |
| 后置条件 | 用户成功登入，进入主界面。 |
| 涉众利益 | 每个游客都可以简单的进行操作 |
| 基本路径 | 1. 系统显示登陆需要的信息。 2. 用户填入登陆需要的信息。 3. 系统检测用户名和密码。 4. 登录成功，进入默认主界面。 |
| 扩展 | 1. 用户输入的信息有错误时，系统提示信息错误。 2. 返回 |
| 字段列表 | 用户名，密码。 |
| 设计规则 | 用户名唯一。 |
| 未解决问题 |  |

|  |  |
| --- | --- |
| 用例名称 | 添加任务 |
| 用例编号 | H003 |
| 执行者 | 系统用户 |
| 前置条件 | 用户已经登陆 |
| 后置条件 | 用户快速添加新的任务。 |
| 涉众利益 | 系统用户可以添加任务 |
| 基本路径 | 1. 系统显示快速添加任务窗口 2. 用户输入任务名称，Enter或者点击添加 3. 系统获得任务名，将任务添加到服务器数据库，同时在主界面收集箱中添加一个任务。 |
| 扩展 |  |
| 字段列表 | 任务名。 |
| 设计规则 | 任务名唯一，默认将任务放在收集箱中。 |
| 未解决问题 |  |

|  |  |
| --- | --- |
| 用例名称 | 编辑任务 |
| 用例编号 | H004 |
| 执行者 | 系统用户 |
| 前置条件 | 用户已经登陆，已经选中需要编辑的任务，进入编辑页面。 |
| 后置条件 | 用户成功完善任务信息，并根据任务信息将任务进行归类。 |
| 涉众利益 | 系统用户可以完善任务信息任务 |
| 基本路径 | 1. 系统显示任务信息编辑窗口 2. 用户输入任务信息，点击保存。 3. 系统获得任务信息，将任务信息添加到服务器数据库，同时在主界面中对任务根据信息进行归类。 4. 进入任务列表界面。 |
| 扩展 | 1. 任务名重复，系统提示错误。 2. 返回。 |
| 字段列表 | 任务名、开始时间，截止时间，备注，情景收集箱，项目收集箱，提醒信息，标签，转发给，优先级，是否完成 |
| 设计规则 |  |
| 未解决问题 |  |

|  |  |
| --- | --- |
| 用例名称 | 删除任务 |
| 用例编号 | H005 |
| 执行者 | 系统用户 |
| 前置条件 | 用户已经登陆，已经选中需要编辑的任务，可以同时删除多个任务。 |
| 后置条件 | 用户成功删除任务，被删除任务存放在垃圾箱中。 |
| 涉众利益 | 系统用户可以删除不在需要的任务。 |
| 基本路径 | 1. 系统显示任务列表。 2. 用户选中需要删除的任务，点击工具栏中删除按钮。 3. 系统将所删除的任务状态更改为“删除”，主界面中将删除的任务存放在垃圾箱中。 |
| 扩展 |  |
| 字段列表 |  |
| 设计规则 | 用户可以采用选中点击删除按钮删除任务，也可以采用右击菜单来删除任务。 |
| 未解决问题 |  |

|  |  |
| --- | --- |
| 用例名称 | 清空垃圾箱 |
| 用例编号 | H006 |
| 执行者 | 系统用户 |
| 前置条件 | 进入垃圾箱面板 |
| 后置条件 | 垃圾箱中的任务在数据库中永久删除，垃圾箱面板中没有任务 |
| 涉众利益 | 用户可以把以后都不需要的任务永久删除 |
| 基本路径 | 1. 系统显示垃圾箱中的任务类表。 2. 用户点击工具栏中的清空回收站按钮。 3. 提示是否永久删除任务。 4. 用户点击确定。 5. 垃圾箱中的任务被永久删除，垃圾箱为空。 |
| 扩展 |  |
| 字段列表 |  |
| 设计规则 |  |
| 未解决问题 |  |

|  |  |
| --- | --- |
| 用例名称 | 添加联系人 |
| 用例编号 | H008 |
| 执行者 | 系统用户 |
| 前置条件 | 进入联系人面板 |
| 后置条件 | 联系人列表增加新的联系人。 |
| 涉众利益 | 用户可以添加联系人， |
| 基本路径 | 1. 用户点击添加联系人按钮。 2. 系统显示所需添加联系人信息。 3. 用户填写联系人信息，点击完成。 4. 系统获得联系人信息，并添加。 5. 在用户联系人类表中增加新添加的联系人。 |
| 扩展 | 1. 如果相同姓名的联系人已经存在，提示错误。 2. 返回 |
| 字段列表 | 联系人姓名、所在群组，备注。 |
| 设计规则 | 联系人姓名必须也是该系统的用户。 |
| 未解决问题 |  |

|  |  |
| --- | --- |
| 用例名称 | 删除联系人 |
| 用例编号 | H009 |
| 执行者 | 系统用户 |
| 前置条件 | 进入联系人面板。 |
| 后置条件 | 选中需要删除的联系人 |
| 涉众利益 | 用户可以删除联系人。 |
| 基本路径 | 1. 系统显示联系人列表。 2. 用户选中需要删除的联系人。 3. 用户点击删除按钮。 4. 系统将该联系人从用户的联系人列表中删除。 |
| 扩展 |  |
| 字段列表 |  |
| 设计规则 |  |
| 未解决问题 |  |

|  |  |
| --- | --- |
| 用例名称 | 完成任务 |
| 用例编号 | H010 |
| 执行者 | 系统用户 |
| 前置条件 | 用户在任意未完成任务列表中。 |
| 后置条件 | 被完成任务进入完成状态，从未完成列表中删除了，进入完成列表 |
| 涉众利益 | 用户可以将任务标记为完成态。 |
| 基本路径 | 1. 系统显示未完成任务列表。 2. 用户选中任务点击完成，或者右击菜单完成。 3. 任务被标记为完成态，从未完成列表中删除，任务进入已完成列表中。 |
| 扩展 | 1. 如果该任务为别人提供的共享任务，则应该通知共享人，任务已完成。 2. 如果该任务自己曾共享给他人，则应该通知被共享人，任务已完成。 |
| 字段列表 |  |
| 设计规则 |  |
| 未解决问题 |  |

## 设计限制（如系统只能采用java语言编写，在eclipse下用等等）

本项目主要采用Java SE完成，所基于的IDE是eclipse。主要利用PSP方法记录和管理过程数据。

系统最多允许50个客户同时连接服务器端。一个账号不能在两个客户端登陆。

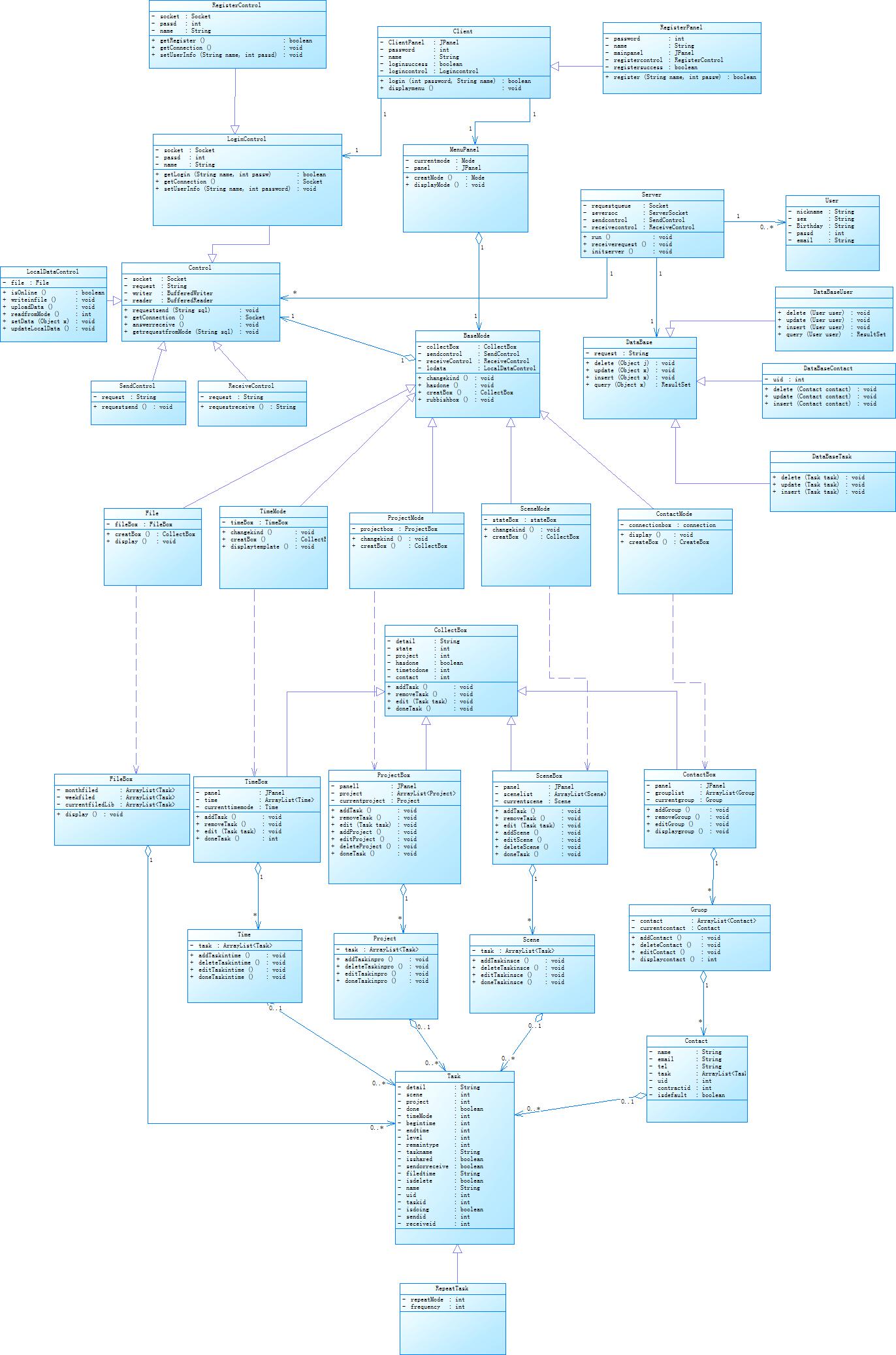
必须实现本地同步，即在断网情况下也能够管理个人事务，再次联网时，进行本地数据和服务器数据的同步。

# 运行环境

本系统主要运行在Windows系统下，要使用系统的完整功能必须在联网情况下。

**系统架构**

## 详细设计类图



## *用户接口设*计（界面设计图例等）

登陆界面：



时间模式下界面设计：



左上角可进行模式选择，选择模式进入对应的界面。

情景模式界面：



回顾模式界面：



界面设计风格基本保持一致，遵循一致性原则；

界面简洁，模块划分清晰，贴近于用户。

## 数据库设计（可以用表或者数据库实体关系图表示）

用户表(User)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 列名 | 说明 | 类型 | 能否为空 | 是否为主键 |
| User\_id | 用户ID | Integer | N | Y |
| User\_name | 用户的用户名 | Varchar(30) | N | N |
| User\_password | 用户的密码 | Varchar(20) | N | N |

用户的个人设置表(Info)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 列名 | 说明 | 类型 | 能否为空 | 是否为主键 |
| Info\_id | 个人设置表ID | Integer | N | Y |
| User\_id | 用户ID | Integer | N | User表 外键 |
| User\_nickname | 用户的昵称 | Varchar(20) | Y | N |
| User\_email | 用户的邮箱地址 | Varchar(30) | Y | N |
| Language | 显示语言(中英文) | Tinyint或枚举  (0或1) | Y(有default值) | N |
| Gender | 性别 | Tinyint或枚举 | Y(有default值) | N |
| Birthday | 生日 | 时间类型 | Y | N |
| Info\_remark | 个人信息备注 | Text | Y | N |
| Info\_photo | 用户照片 | longblob | Y | N |

用户的任务表(Task表)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 列名 | 说明 | 类型 | 能否为空 | 是否为主键 |
| Task\_id | 任务id号 | Integer | N | Y |
| Task\_name | 任务名 | Varchar(50) | N | N |
| Task\_information | 任务的详细备注信息 | Text | Y(默认为NULL) | N |
| User\_id | 用户ID | Integer | N | User表 外键 |
| Project\_name | Projet是否取默认值，默认为null,如果值不为null，则加上用户自己新建的项目名称 | string | Y(有default值)  默认为null | Project表 外键 |
| Task\_scene | Scene(情景)是否取默认值，默认为null,如果值取1，则在程序中转向查找scene表 | 枚举(0,1) | Y(有default值)  默认为null即0 | N |
| Task\_priority | Task优先级 | 枚举(分为0,1,2)即高中低优先级 | Y(有default值)  默认为低即0 | N |
| Task\_repeat | 可否重复，如果值为1，则在程序中转向查找repeattask表 | Tinyint或枚举(0或1) | Y(有default值)  默认为不重复即0 | N |
| Task\_shared | 任务是否是与其他的用户共享的，默认值为0，如果值取1，则在程序中转向查找sharedtask表 | 枚举(0,1) | Y(有default值)  默认为0，即不共享 | N |
| Task\_starttime | 任务开始时间 | 时间类型 | Y(有default值)  读取当时的当地时间 | N |
| Task\_endtime | 任务完成时间 | 时间类型 | Y(有default值)  今天的截止时间 | N |
| Isdoing | 任务是否正在进行 | bool | Y(有default值) | N |
| Isdone | 任务是否已经完成 | bool | Y(有default值) | N |
| Isdelete | 任务是否被转移到了垃圾箱中 | bool | Y(有default值) | N |

项目模式(Project表) 即由用户自己新建的项目，默认的项目无需在此考虑

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 列名 | 说明 | 类型 | 能否为空 | 是否为主键 |
| Project\_id | 项目ID | Integer | N | Y |
| Project\_name | 新建的项目名称，具有唯一性 | string | N(在程序中设定默认为newproject+id) | N |
| User\_id | 用户ID | Integer | N | User表 作为外键 |

情景模式表(Scene表)即5个情景和用户自己新建的情景，默认的情景为null

在建立数据库的时候必须同时事先建立5个情景模式，即家里，办公室，电脑，电话，外出，这样可以避免区分默认的5个场景和用户自己新建的情景。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 列名 | 说明 | 类型 | 能否为空 | 是否为主键 |
| Scene\_id | 情景模式ID | Integer | N | Y |
| Scene\_name | 情景模式名称 | Varchar(30) | N | N |
| Task\_id | 原任务ID号 | Integer | N | task表 作为外键 |
| Task\_name | 任务名 | Varchar(50) | N | task表 作为外键 |
| User\_id | 用户ID | Integer | N | User表 作为外键 |

多个用户共同的任务表(Sharedtask表)，由他人转发给你的任务，你不能再转发给别人。

另外考虑到任务在转发给他人从而成为共同的任务时，这些任务都会转入发送方和接收方的等待目录下，故而添加isdone,isdong,isdelete属性以方便对于这些共享任务的处理。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 列名 | 说明 | 类型 | 能否为空 | 是否为主键 |
| Sharedtask\_id | 共享的任务ID，和原任务表的ID号是一对一的关系 | Integer | N | Y |
| Task\_id | 原任务ID号 | Integer | N | task表 作为外键 |
| Task\_name | 任务名 | Varchar(50) | N | task表 作为外键 |
| sender\_id | 发起人的用户ID | Integer | N | User表 作为外键 |
| Resever\_id | 接受人的用户ID | Integer | N | User表 作为外键 |
| Isdoing | 任务是否正在进行 | bool | Y(有default值) | User表 作为外键 |
| Isdone | 任务是否已经完成 | bool | Y(有default值) | User表 作为外键 |
| Isdelete | 任务是否被转移到了垃圾箱中 | bool | Y(有default值) | User表 作为外键 |

重复的任务表(repeattask表)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 列名 | 说明 | 类型 | 能否为空 | 是否为主键 |
| Repeattask\_id | 重复的任务表的ID | Integer | N | Y |
| Task\_id | 和原任务表的ID号是一对一的关系 | Integer | N | task表 作为外键 |
| Task\_name | 任务名 | Varchar(50) | N | task表 作为外键 |
| User\_id | 用户ID | Integer | N | User表 作为外键 |
| Repeat\_type | 重复任务的类型 | 枚举(分别用0,1,2……来表示是按日，周，工作日，月，年来重复相应的任务)，再在程序中按所得的Rapeat\_type的值，然后在程序中决定跳转的方向。 | N | N |

重复任务的类型

共需建5个表，Repeatbyday表，Repeatbyweek表，Repeatbywookday表，Repeatbymooth表，Repeatbyyear表

Repeatbyday表

到时开始时间是读取当时的系统时间。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 列名 | 说明 | 类型 | 能否为空 | 是否为主键 |
| Day\_frequency | 重复频率 | Integer | N(默认为1) | N |
| Canbe\_end | 是否设定结束时间 | Bool | N(默认为false，即从不结束) | N |
| end\_time | 结束时间 | 时间类型 | 如果Canbe\_end为false，则为null，如果Canbe\_end为true，则加上结束时间，这点在程序中加以控制。 | N |
| Task\_id | 原任务ID号 | Integer | N | task表 作为外键 |
| Task\_name | 任务名 | Varchar(50) | N | task表 作为外键 |
| User\_id | 用户ID | Integer | N | User表 作为外键 |

Repeatbyweek表

到时开始时间是读取当时的系统时间。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 列名 | 说明 | 类型 | 能否为空 | 是否为主键 |
| Week\_ frequency | 重复频率 | Integer | N(默认为1) | N |
| Repeatday\_rate | 重复日的设定，为避免存储过于复杂，将其设为long int型，如1010101表示周日，周二，周四，周六是被选中的，在程序中可以通过相除来实现这一算法思想。 | Long int | N | N |
| Canbe\_end | 是否设定结束时间 | Bool | N(默认为false，即从不结束) | N |
| end\_time | 结束时间 | 时间类型 | 如果Canbe\_end为false，则为null，如果Canbe\_end为true，则加上结束时间，这点在程序中加以控制。 | N |
| Task\_id | 原任务ID号 | Integer | N | task表 作为外键 |
| Task\_name | 任务名 | Varchar(50) | N | task表 作为外键 |
| User\_id | 用户ID | Integer | N | User表 作为外键 |

Repeatbywookday表

到时开始时间是读取当时的系统时间。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 列名 | 说明 | 类型 | 能否为空 | 是否为主键 |
| Canbe\_end | 是否设定结束时间 | Bool | N(默认为false，即从不结束) | N |
| end\_time | 结束时间 | 时间类型 | 如果Canbe\_end为false，则为null，如果Canbe\_end为true，则加上结束时间，这点在程序中加以控制。 | N |
| Task\_id | 原任务ID号 | Integer | N | task表 作为外键 |
| Task\_name | 任务名 | Varchar(50) | N | task表 作为外键 |
| User\_id | 用户ID | Integer | N | User表 作为外键 |

Repeatbymooth表

到时开始时间是读取当时的系统时间。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 列名 | 说明 | 类型 | 能否为空 | 是否为主键 |
| mooth\_frequency | 重复频率 | Integer | N(默认为1) | N |
| Canbe\_end | 是否设定结束时间 | Bool | N(默认为false，即从不结束) | N |
| end\_time | 结束时间 | 时间类型 | 如果Canbe\_end为false，则为null，如果Canbe\_end为true，则加上结束时间，这点在程序中加以控制。 | N |
| Repeat\_mtype | 重复标准，是按日期还是星期几 | 枚举(0或1) | N(默认值为0，即按日期) | N |
| Task\_id | 原任务ID号 | Integer | N | task表 作为外键 |
| Task\_name | 任务名 | Varchar(50) | N | task表 作为外键 |
| User\_id | 用户ID | Integer | N | User表 作为外键 |

Repeatbyyear表

到时开始时间是读取当时的系统时间。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 列名 | 说明 | 类型 | 能否为空 | 是否为主键 |
| year\_frequency | 重复频率 | Integer | N(默认为1) | N |
| Canbe\_end | 是否设定结束时间 | Bool | N(默认为false，即从不结束) | N |
| end\_time | 结束时间 | 时间类型 | 如果Canbe\_end为false，则为null，如果Canbe\_end为true，则加上结束时间，这点在程序中加以控制。 | N |
| Task\_id | 原任务ID号 | Integer | N | task表 作为外键 |
| Task\_name | 任务名 | Varchar(50) | N | task表 作为外键 |
| User\_id | 用户ID | Integer | N | User表 作为外键 |

用户的联系人列表(contract)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 列名 | 说明 | 类型 | 能否为空 | 是否为主键 |
| Contract\_id | 联系人ID号，一个联系人对应一个用户ID | Integer | N | Y |
| User\_id | 用户ID | Integer | N | User表 外键 |
| Contract\_name | 被用户用来备注联系人使用的名字 | Varchar(30) | Y | N |
| group\_isdefault | 被用户是否放在默认的group表中，默认为0，如果值取1，则在程序中转向查找group表 | 枚举(0,1) | Y(有default值)  默认为null即0 | N |
| Isonline | 是否在线 | Tinyint或枚举 | N(因为是由服务器给予的信息) | N |

用户新建的联系人组群(group表)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 列名 | 说明 | 类型 | 能否为空 | 是否为主键 |
| Group\_id | 组群ID号 | Integer | N | Y |
| Group\_name | 组群名称 | Varchar(30) | N | N |
| Contract\_id | 联系人ID号 | Integer | N | contract表 外键 |
| Contract\_name | 被用户用来备注联系人使用的名字 | Varchar(30) | Y | contract表 外键 |
| User\_id | 用户ID | Integer | N | User表 外键 |

## 类的接口设计（各个类在编写是必须服从那些接口限制）

Client 的login方法作为与LoginControl交互的接口。

LoginControl中SetUserInfo是对Client提供的接口（传入账号密码）。

LoginControl 中的getLogin是与Server交互接口。

Control中的getRequestFromMode 提供与BaseMode交互接口，requestsend,answerreceive是提供了与Server交互接口。

Server中的receiverequest是与Control交互的接口。

DataBase提供了delete,update,insert,query方法作为外部调用数据库操作的接口。

# 参考文档

《