<项目名称>

**错误！未定义书签。**

版本 <1.0>

修订历史记录

|  |  |  |  |
| --- | --- | --- | --- |
| **日期** | **版本** | **说明** | **作者** |
| <日/月/年> | <x.x> | <详细信息> | <姓名> |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

目录

1. 简介 4

1.1 目的 4

1.2 参考资料 4

2. 用例视图 4

3. 逻辑视图 4

3.1 概述 4

3.2 在构架方面具有重要意义的设计包 4

4. 进程视图 4

5. 部署视图 4

6. 实现视图 5

7. 数据视图（可选） 5

8. 核心算法设计（可选） 5

软件架构文档 （简化版）

# 简介

## 目的

本文档将从构架方面对系统进行综合概述，其中会使用多种不同的构架视图来描述系统的各个方面。它用于记录并表述已对系统的构架方面作出的重要决策。

# 用例视图

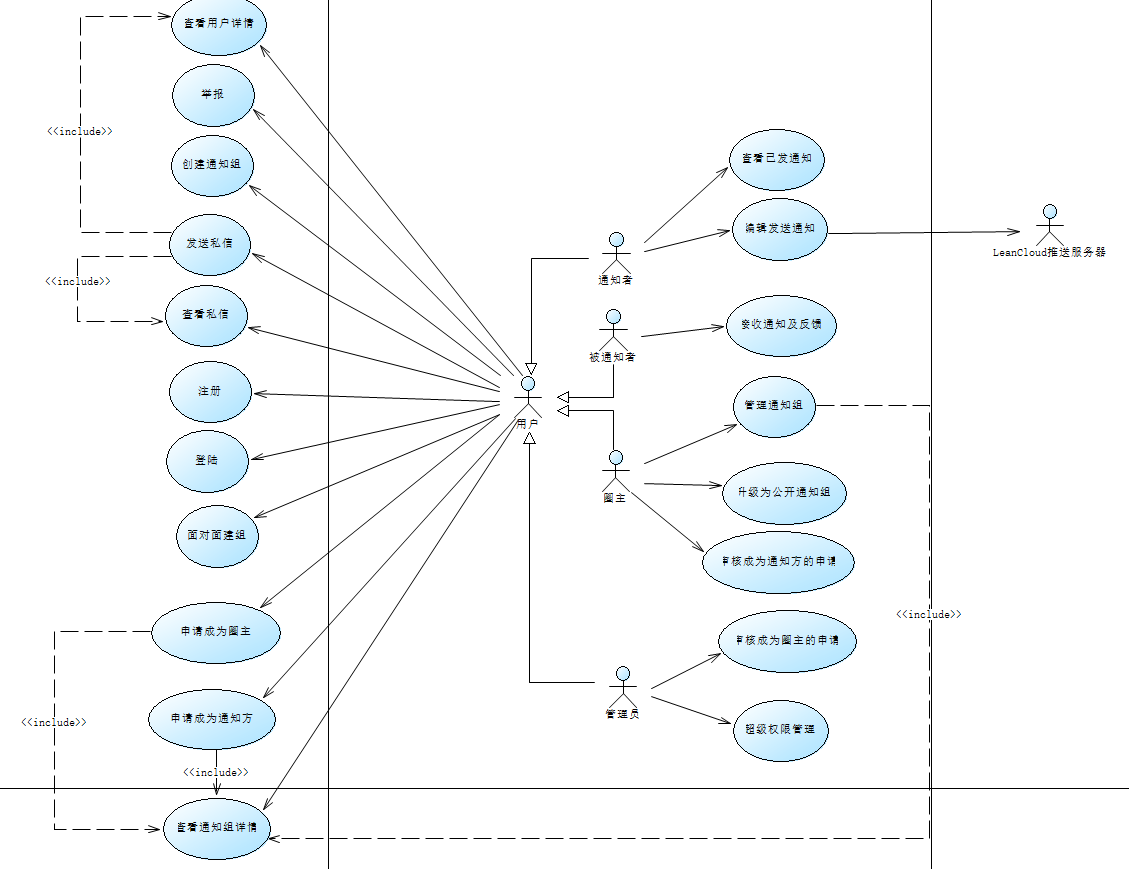


图1.用例视图

# 逻辑视图

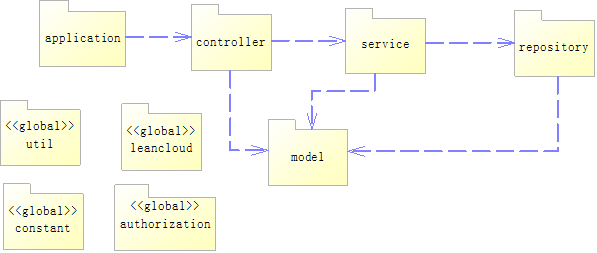


图2.服务器端逻辑视图

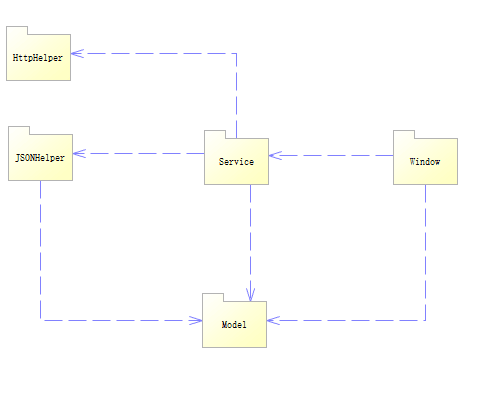


图3.windows端逻辑视图

## 概述

（1）服务器端：

整体分为application、controller、service、repository、model、leancloud、util、constant、authorization这几个逻辑上的包。

另外controller service repository model四个包中的类可以分为如下几个功能性模块：

user：用户信息

house：通知组信息

member：通知组成员信息

raven：通知信息

serve：用户申请信息

whistleblowing：举报信息

（2）Windows端：

用户在Window上进行操作，由Window调用Service层中的方法。Service层中的方法将操作分解为一个或多个基本操作，通过HttpHelper向服务器发送请求调用REST API。服务器返回的JSON字符串被JSONHelper通过解析成对象，返回到Window，最终将信息呈现在用户前。

## 在构架方面具有重要意义的设计包

controller：作为和客户端进行交互的接口，向外暴露restful api，在客户端和服务端间传送数据，并作基本的输入校验

service：业务的主要实现处。藉由repository层提供的原子操作完成业务逻辑并将结果返回给controller层

repository：根据Model将数据库操作包装为原子业务逻辑供上层使用

leancloud ：实现即时通信和消息推送的第三方库

authorization：借助api token实现比session更为完善的用户身份验证功能

model：指示了业务数据在数据库中的储存形式，也作为查询和修改的基本单位

# 部署视图

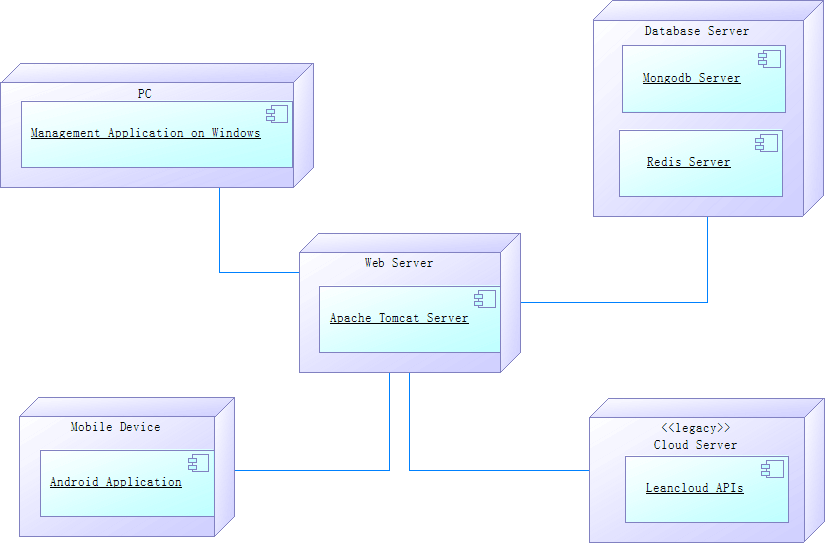


图4.部署图

# 实现视图

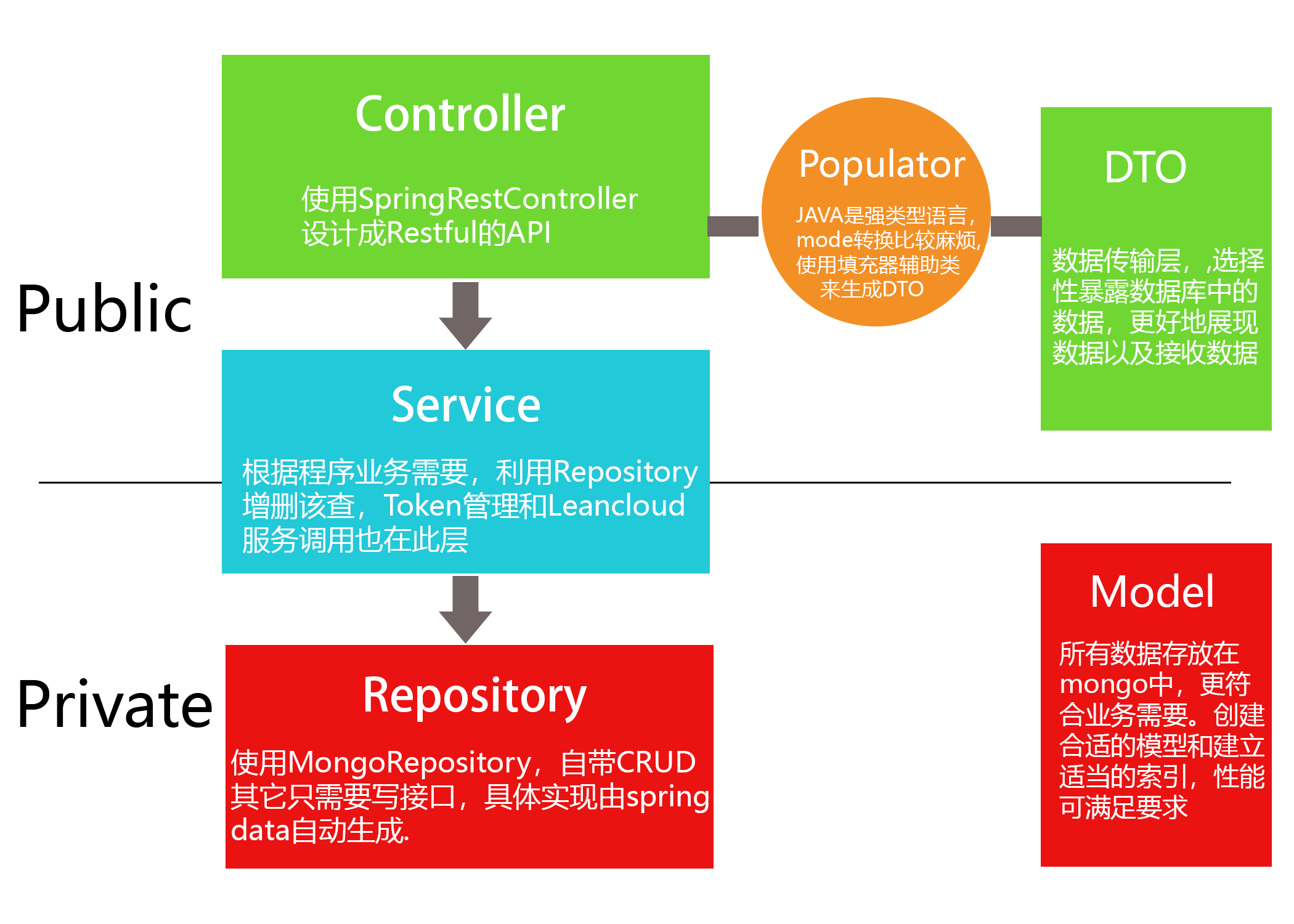


图5.服务器端实现视图

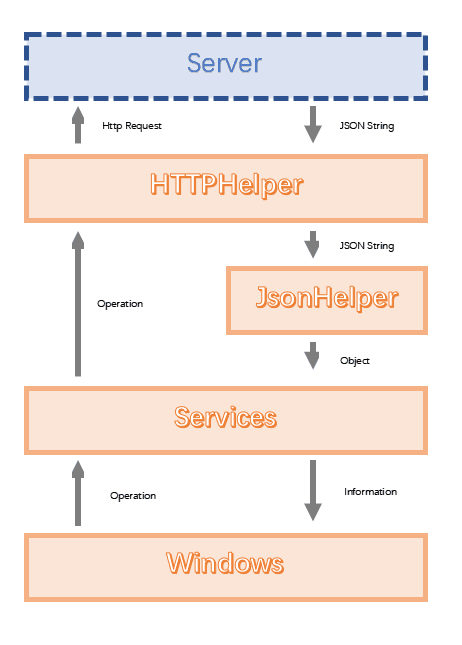


图6.windows端实现视图