智能有声书

软件架构文档

版本 <1.0>

修订历史记录

|  |  |  |  |
| --- | --- | --- | --- |
| **日期** | **版本** | **说明** | **作者** |
| 2019/07/04 | 1.1 |  | 徐轶喆 |
| 2019/07/10 | 1.2 |  | 徐轶喆 |
|  |  |  |  |

**目录**

第1级 1

第2级 2

第2级 3

第1级 4

第2级 5

第2级 6

**软件架构文档**

# 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。

架构设计定义了项目软件的全貌，记录了最重要的设计决策，并成为项目软件随后的详细设计与实现工作的战略指导原则。而在本文档中，我们主要使用视图来描述和归档软件的高低层次、表示软件架构的某个方面、展示软件系统的某些特性。

本文一共列举了个视图，分别用于展示小组项目在架构设计当中对架构设计的不同方面采用的不同策略。我们的项目设计从分析需求、设计用例，到设计具体逻辑架构，再到分进程、开发架构的方向进行设计，最后设计出项目软件的物理架构，生成具体的部署与实现视图。

## 参考资料

1. 《软件工程原理》.作者:沈备军、陈昊鹏、陈雨亭.高等教育出版社.2014.12版
2. 《UML视图种类划分》.作者:sduliding.2015.09.

<https://blog.csdn.net/sduliding/article/details/48246713>

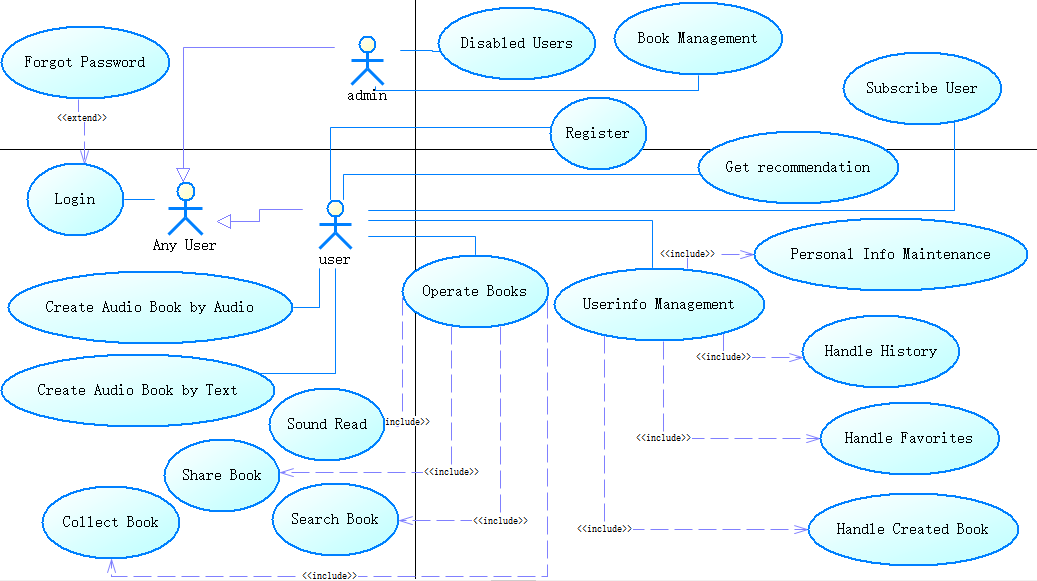
1. 《UML物理视图》.作者:只是为了记录一刻的所得.2013.08.07

<https://blog.csdn.net/seacean2000/article/details/10374573>

1. 《“4+1”视图模型》.作者:QI XIN.2013.06.30

<https://www.cnblogs.com/qixin/p/3163536.html>

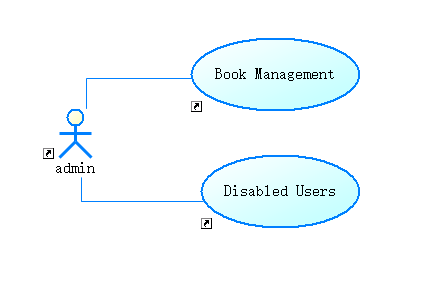
# 用例视图



完整的用例图 图(1)

我们小组将只能有声书的用例模型分为了6个包，每个包都有一张具体用例图，分别代表了不同的场景。下面的内容将会一一列举这些场景和用例。

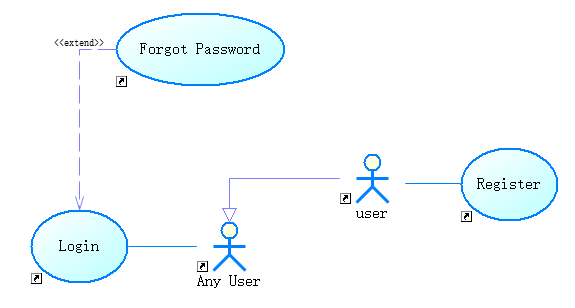
**场景1: Admin Management**



包Admin Management用例图 图(2)

Admin Management用例描述了管理员用户想要禁用、解禁一位普通用户，或管理有声书的场景和需求。

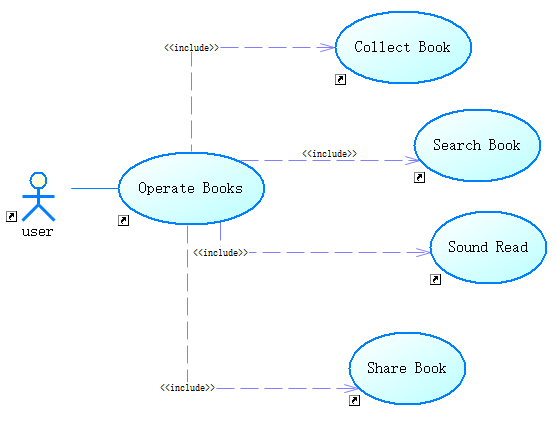
**场景2: User Recognition**



包 User Recognition用例图 图(3)

User Recognition用例描述了任意用户想要登录有声书系统(包括普通用户系统和管理员系统)，或普通用户想要注册普通用户系统账号的场景和需求。

**场景3: User Book Management**

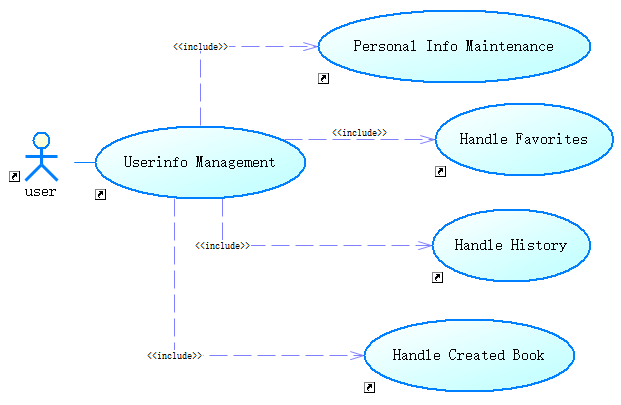


包 User Book Management用例图 图(4)

User Book Management用例描述了普通用户想要对有声书进行操作的场景和需求。

普通用户对有声书的操作包括了4个子用例，分别是收藏有声书、搜索有声书、有声书阅读、分享有声书。

**场景4: User Information Management**

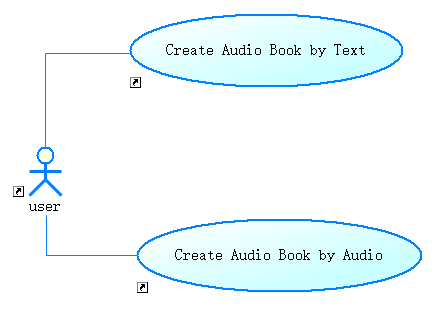


包 User Information Management用例图 图(5)

User Information Management用例描述了普通用户想要管理个人信息的场景和需求。

普通用户管理个人信息包括4个子用例，分别是修改个人信息、管理收藏夹、管理阅读历史记录、管理创作的有声书。

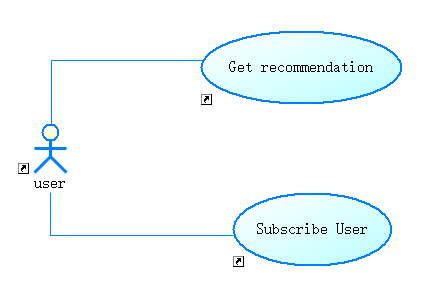
**场景5: User Book Creation**



包 User Book Creation用例图 图(6)

User Book Creation用例描述了普通用户想要通过文本文件、或音频文件创作有声书的场景和需求。

**场景6: User Transaction**



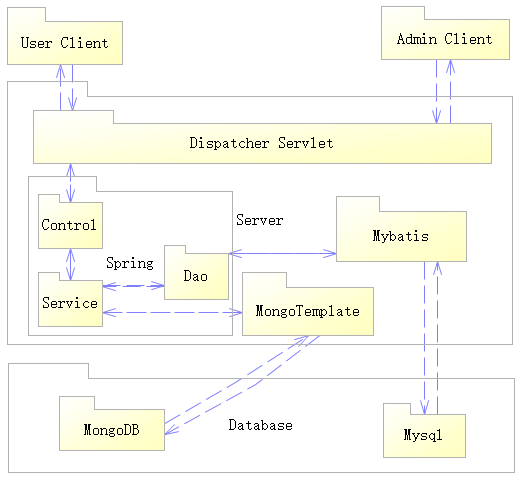
包 User Transaction用例图 图(7)

User Transaction用例描述了普通用户获取推荐有声书、关注其他用户的场景和需求。

# 逻辑视图

## 概述

逻辑架构设计方面我们采用三层架构风格。



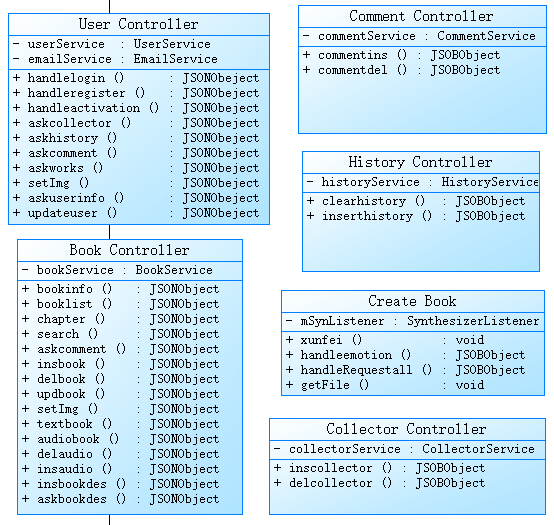
逻辑架构图 图(8)

表示层包含普通用户App移动端和管理员系统客户端，向服务层发送请求。

服务层通过Dispatcher Servlet包响应客户端发送的请求，传递给Spring类包，经由控制类调用相应的服务类，再由服务类调用数据访问与缓存类，最终实现对数据层的访问。

## 在构架方面具有重要意义的设计包

* + 1. **Control Package**



Control包类图 图(9)

Control包中包含了各种用于接收对应不同实体操作请求的控制类，如图。下面将对于其中每个类的具体功能进行描述。

**User Controller:**

一个UserService类私有变量，注解为自动装配，用于调用Service包中UserService类的各个接口。

一个EmailService类私有变量，注解为自动装配，用于调用Service包中EmailService类的各个接口。

setRHeader函数，用于设置响应头。

handlelogin函数，用于接收普通用户的登录请求，并调用普通用户的登录服务。

handleregister函数，用于接收普通用户注册的请求，并调用普通用户注册服务。

handleactivation函数，用于接收普通用户邮箱验证激活账号的请求，并调用普通用户账号激活服务。

askcollector函数，用于接收普通用户查看收藏夹的请求，并调用普通用户查看收藏夹服务。

askhistory函数，用于接收普通用户查看阅读历史记录的请求，并调用普通用户查看阅读历史记录服务。

askcomment函数，用于接收普通用户查看评论记录的请求，并调用普通用户查看评论记录服务。

askworks函数，用于接收普通用户查看创建的有声书的请求，并调用普通用户查看创建的有声书服务。

**Book Controller:**

一个BookService类私有变量，注解为自动装配，用于调用Service包中BookService类的各个接口。

bookifo函数，用于接收普通用户查看有声书信息的请求，并调用查询有声书信息服务。

booklist函数，用于接收查看所有有声书的请求，并调用查询所有有声书列表服务。

chapter函数，用于接收查看一本有声书某一个章节的请求，并调用查询有声书章节服务。

search函数，用于接收通过书名查找有声书的请求，并调用通过标题查询有声书服务。

askcomment函数，用于接收查看有声书评论的请求，并调用查询有声书评论服务。

insbook函数，用于接收添加一本有声书的请求，并调用插入一本有声书服务。

delbook函数，用于接收删除一本有声书的请求，并调用删除一本有声书服务。

updbook函数，用于接收更新一本有声书的请求，并调用更新一本有声书服务。

setImg函数，用于接收设置一本有声书封面的请求，并调用插入一本有声书的封面服务。

textbook函数，用于接收的请求，并调用服务。

audiobook函数，用于接收查看一本有声书音频的请求，并调用查询有声书音频服务。

delaudio函数，用于接收删除一本有声书音频的请求，并调用删除有声书音频服务。

insaudio函数，用于接收添加一本有声书音频的请求，并调用插入有声书音频服务。

insbookdes函数，用于接收添加一本有声书简介的请求，并调用插入有声书简介服务。

askbookdes函数，用于接收查看一本有声书简介的请求，并调用查询有声书简介服务。

**Comment Controller:**

一个CommentService类私有变量，注解为自动装配，用于调用Service包中CommentService类的各个接口。

commentins函数，用于接收普通用户评论的请求，并调用插入一条评论服务。

commentdel函数，用于接收删除一条评论的请求，并调用删除一条评论服务。

**Collector Controller:**

一个CollectorService类私有变量，注解为自动装配，用于调用Service包中CollectorService类的各个接口。

inscollector函数，用于接收普通用户将一本有声书加入收藏夹的请求，并调用向收藏夹插入一本有声书服务。

delcollector函数，用于接收普通用户将一本有声书从收藏夹删除的请求，并调用删除收藏夹内一本有声书服务。

**History Controller:**

一个HistoryService类私有变量，注解为自动装配，用于调用Service包中HistoryService类的各个接口。

clearhistory函数，用于接收删除一本有声书的阅读历史记录的请求，并调用删除一本有声书的阅读历史记录服务。

inserthistory函数，用于接收添加一本有声书的阅读历史记录的请求，并调用添加一本有声书的阅读历史记录服务。

**Create Book:**

一个SynthesizerListener的私有变量，

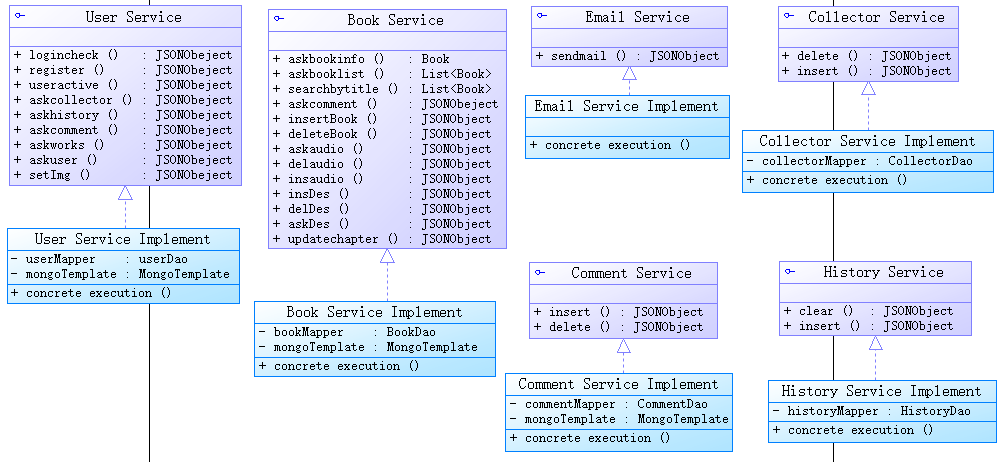
xunfei函数，用于

handleemotion函数，用于

handleRequetall函数，用于

getFile函数，用于

* + 1. **Service Package**

****

Service包类图 图(10)

Service包中包含了各种对应各个实体类的不同的服务类，用于调用并处理数据，返回给控制层响应body，如图。下面将对于其中每个类的具体功能进行描述。

**User Service:**

一个UserDao私有变量，注解为自动装配，用于调用Dao包中UserDao类的各个接口。

一个MongoTemplate私有变量，注解为自动装配，用于调用mongoDB操作的一些接口。

logincheck函数，调用查询用户数据的数据访问层接口，返回信息是否通过查询。

register函数，调用插入注册用户数据的数据访问层接口，返回是否成功。

useractive函数，调用设置用户激活状态的数据访问层接口，返回是否成功。

askcollector函数，调用查询用户收藏夹的数据访问层接口，返回书籍列表。

askhistory函数，调用查询用户阅读历史的数据访问层接口，返回书籍列表。

askcomment函数，调用查询用户的数据访问层接口和查找用户评论的mongoDB接口，返回评论实体列表。

askworks函数，调用查询用户创建有声书的数据访问层接口，返回书籍列表。

askuser函数，调用查询用户具体信息的数据访问层接口和查询用户头像的mongoDB接口，返回用户实体。

setImg函数，调用插入用户头像的mongoDB接口，返回是否成功。

**Book Service:**

一个BookDao私有变量，注解为自动装配，用于调用Dao包中BookDao类的各个接口。

一个MongoTemplate私有变量，注解为自动装配，用于调用mongoDB操作的一些接口。

askbookinfo函数，调用查询一本有声书信息的数据访问层接口，返回书籍实体。

askbooklist函数，调用查询所有有声书列表的数据访问层接口，返回书籍列表。

searchbytitle函数，调用查询相似标题有声书的数据访问层接口，返回书籍列表。

askcomment函数，调用查询一本有声书评论的数据访问层接口，返回评论列表。

insertBook函数，调用插入一本有声书的数据访问层接口，不返回。

deleteBook函数，调用删除一本有声书的数据访问层接口，不返回。

askaudio函数，调用查询一本有声书音频的mongoDB接口，返回音频。

delaudio函数，调用删除一本有声书音频的mongoDB接口，不返回。

insaudio函数，调用插入一本有声书音频的mongoDB接口，不返回。

insDes函数，调用插入一本有声书简介的mongoDB接口，不返回。

delDes函数，调用删除一本有声书简介的mongoDB接口，不返回。

askDes函数，调用查询一本有声书简介的mongoDB接口，返回书籍的简介。

updatechapter，函数，调用更新一本有声书章节的mongoDB接口，返回是否成功。

**Comment Service:**

一个CommentDao私有变量，注解为自动装配，用于调用Dao包中CommentDao类的各个接口。

一个MongoTemplate私有变量，注解为自动装配，用于调用mongoDB操作的一些接口。

delete函数，调用删除一条评论的数据访问层接口和mongoDB接口，返回是否成功。

insert函数，调用插入一条评论的数据访问层接口和mongoDB接口，返回是否成功。

**Collector Service:**

一个CollectorDao私有变量，注解为自动装配，用于调用Dao包中CollectorDao类的各个接口。

insert函数，调用向收藏夹插入一本有声书的数据访问层接口，返回是否成功。

delete函数，调用向收藏夹删除一本有声书的数据访问层接口，返回是否成功。

**History Service:**

一个HistoryDao私有变量，注解为自动装配，用于调用Dao包中HistoryDao类的各个接口。

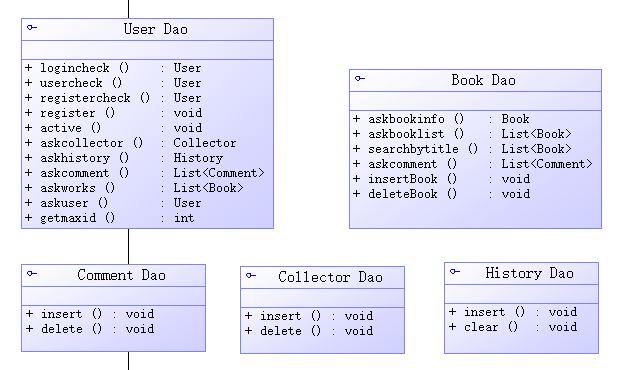
clear函数，调用清除所有阅读有声书历史记录的数据访问层接口，返回是否成功。

insert函数，调用插入一条阅读有声书历史记录的数据访问层接口，返回是否成功。

**Email Service:**

sendmail函数，调用javax.mail中的接口生成激活邮件并发送，返回为空。

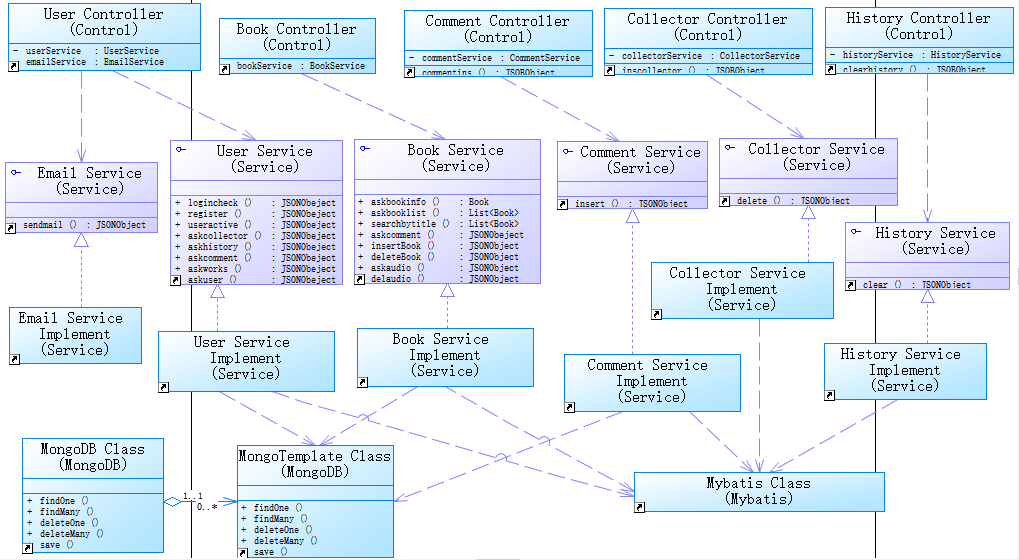
* + 1. **Dao Package**



Dao包类图 图(11)

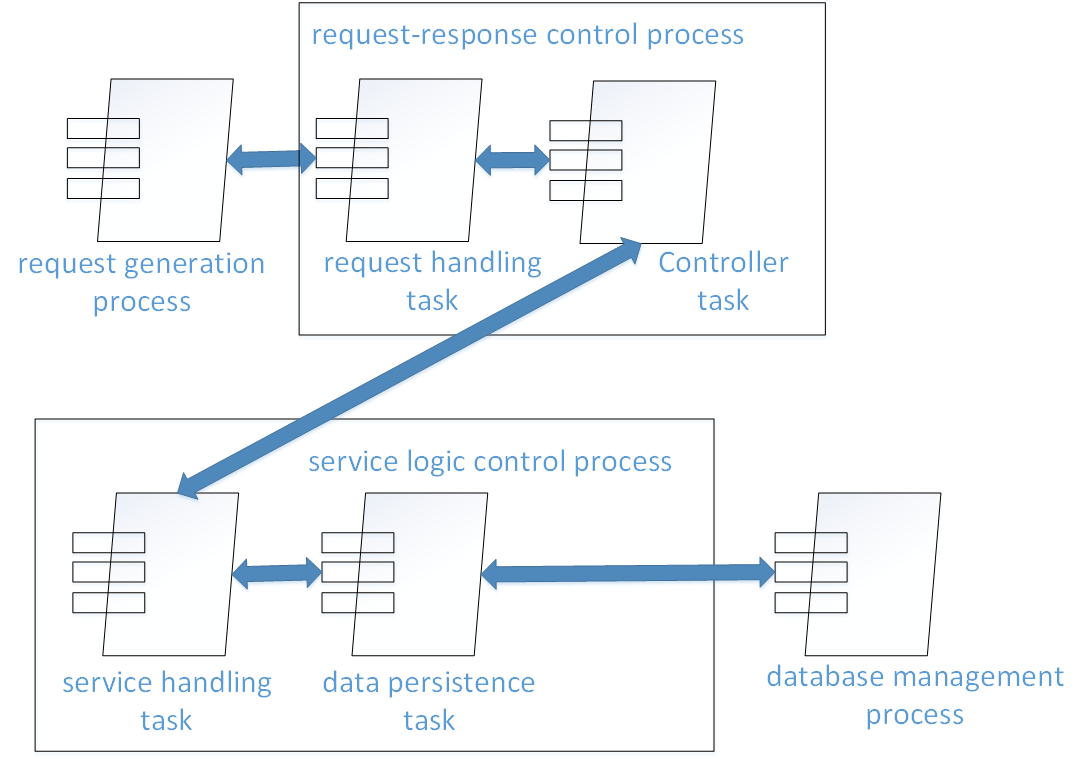
Dao包中包含了各种对应各个实体类的数据访问和存储逻辑，并将结果返回给服务层。对数据库访问使用Mybatis框架，在包中，一个实体类对应一个Dao interface和一个XML配置文件。在interface定义了各种需要的实例，通过MAPPER XML自动生成。

## 在spring包内各个类的调用关系



Spring包类关联图 图(12)

# 进程视图



进程视图 图(13)

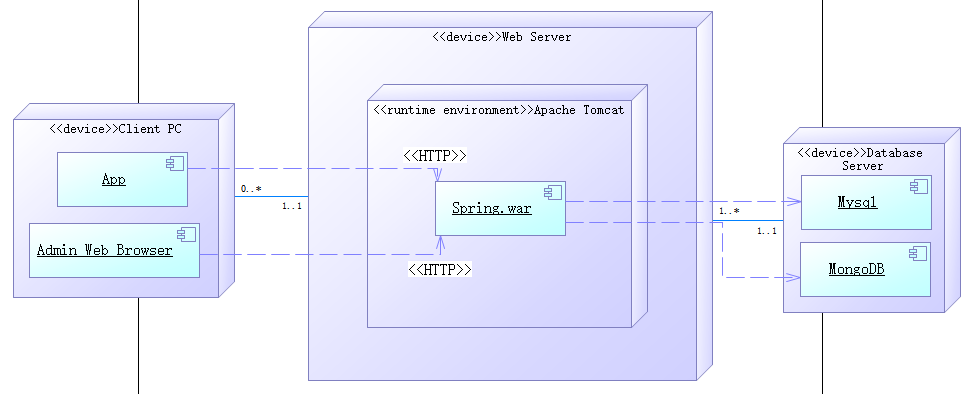
request generation进程为用户产生请求的进程组，属于轻量级进程，与下一级进程的通信为消息汇合。

request-response control进程为响应请求的控制进程，属于多个轻量级进程组成的重量级进程，与下一级进程的通信为消息传递。该进程包含两个任务项，分别负责接收用户请求和调用相应的服务。

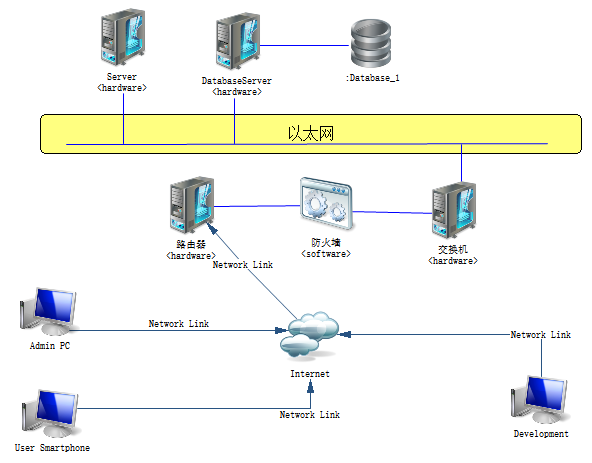
service logic control进程为业务服务逻辑的控制进程，属于多个轻量级进程组成的重量级进程，与下一级进程的通信为消息传递。该进程包含两个任务项，分别负责业务实现和数据的持久化。

database management进程为数据库管理进程，属于轻量级进程。

# 部署视图



部署视图 图(14)



具体网络视图 图(15)

软件的物理架构如上图所示。

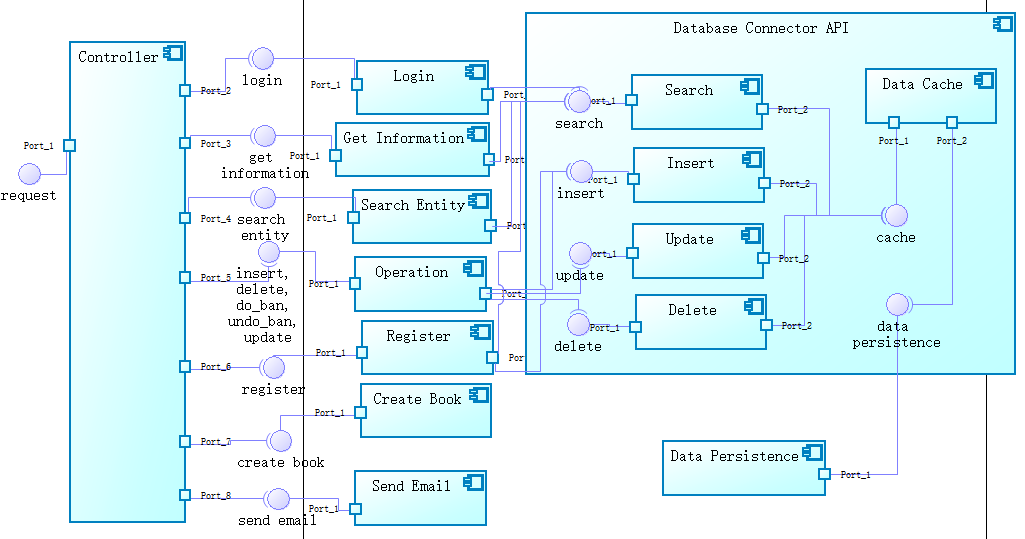
客户端节点上部署了普通用户的App移动端和管理员用户的浏览器客户端。各机器设备间相互独立，通过Internet连接到服务器，对应到进程视图中请求生成进程。

服务器节点上部署了Tomcat运行环境，在其中部署了后端代码集成的.war文件。考虑到该App服务的访问量不会很大，因此将请求的处理控制进程和业务逻辑控制进程部署在同一台服务器上。

服务器节点和数据库服务器节点通过以太网连接。

数据库服务器节点上部署了Mysql和MongoDB数据库管理系统，对应于进程视图中的数据库管理进程。

# 实现视图



构件图 图(16)

上图为软件的构件图，展示了软件实现模型和架构设计中一些重要的构件。

软件实现模型中，还是将后端分为控制、服务、数据三层，数据库管理系统定为子系统。

Controller构件提供接受前端请求的接口，并调用服务层各个构件。

Login构件提供登录服务接口，并调用数据层的数据查找接口。

Get Information构件提供获取各个实体类单个实体详细信息的接口，并调用数据层的数据查找接口。

Search Entity构件提供查找符合相应条件实体集的接口，并调用数据层的数据查找接口。

Operation构件提供对各类实体进行操作的接口，并调用数据层的插入、更新、删除接口。

Register构件提供普通用户注册服务的接口，并调用数据层的查找、插入接口。

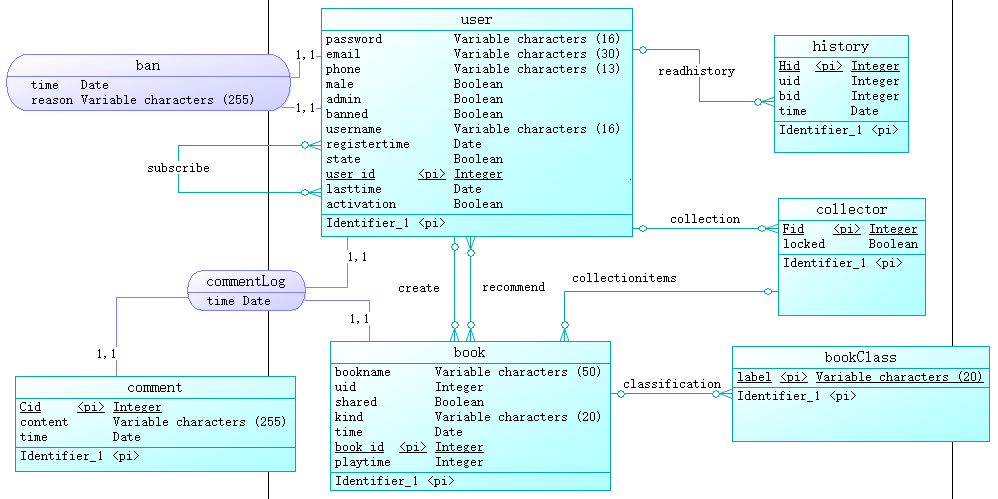
Create Book构件提供创建有声书音频的接口。

Send Email构件提供发送验证邮件服务的接口。

Data Cache构件提供数据缓存接口，起到缓存数据的作用。

Data Persistence构件为数据持久化提供实现接口，由数据库管理子系统实现。

# 数据视图



数据视图 图(17)

上图为软件的概念模型，实体类和关联类，以及类与类之间n对n关系都如图所示。

# 核心算法设计