<阅读规划与管理系统>

软件架构文档

版本 <1.0>

修订历史记录

|  |  |  |  |
| --- | --- | --- | --- |
| **日期** | **版本** | **说明** | **作者** |
| 2016.7.23 | 1.0 | 软件架构信息 | 杨帆、曾鹤鸣、凌康伟 |
|  |  |  |  |
|  |  |  |  |
|  |  |  |  |

目录

1. 简介 2

1.1 目的 2

1.2 参考资料 2

2. 用例视图 2

3. 逻辑视图 2

3.1 概述 2

3.2 在构架方面具有重要意义的设计包 2

4. 进程视图 2

5. 部署视图 2

6. 实现视图 2

7. 数据视图（可选） 2

8. 核心算法设计（可选） 2

软件架构文档 （简化版）

# 简介

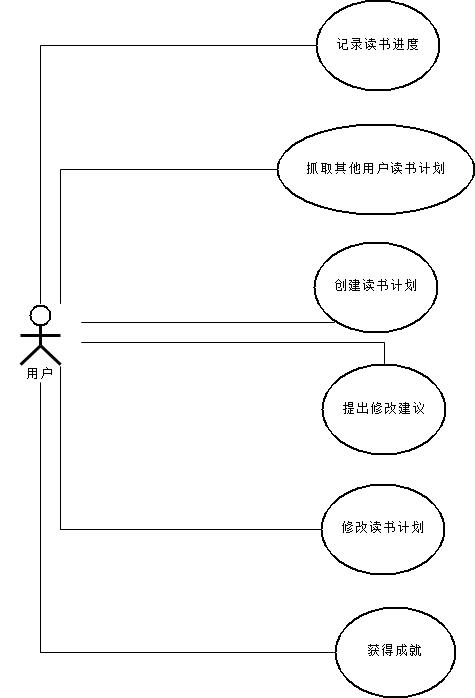
## 目的

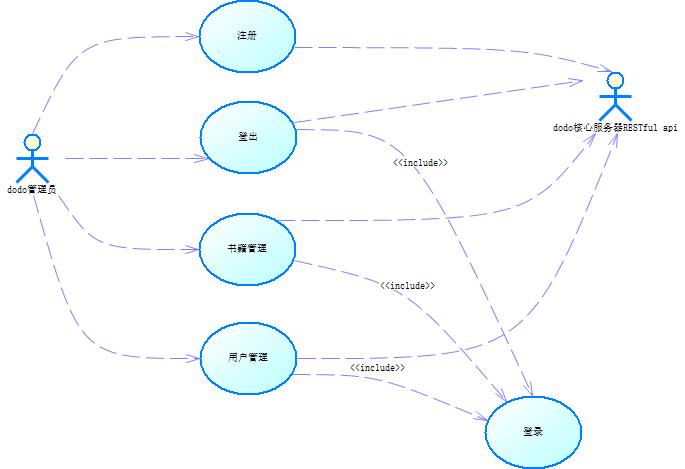
本文档将从构架方面对系统进行综合概述，其中会使用多种不同的构架视图来描述系统的各个方面。它用于记录并表述已对系统的构架方面作出的重要决策。

## 参考资料

《软件工程原理》

# 用例视图





## 记录阅读进度：

基本流：

1. 用户进入读书计划界面。
2. 用户每次阅读后，记录其阅读到多少页。
3. 系统将用户阅读进度信息保存到数据库中。

## 抓取其他用户阅读计划

基本流：

1. 用户进入读书计划搜索界面。
2. 用户点击搜索框，输入关键词进行搜索。
3. 系统向用户展示搜索结果。
4. 用户点击添加，将该读书计划抓取下来

备选流：

3a.若有合法的搜索结果，则用户可以将其添加。

3b.若搜索无合法结果，则搜索框展示内容为空。

## 创建读书计划

基本流：

1. 用户点击创建读书计划按钮。
2. 用户在新建框中点击添加读书的阶段。
3. 在每个读书阶段里，用户可以添加一本或多本书。
4. 用户为每个阶段设置完成该阶段所需要读的最少的数量以及完成该阶段可以获得的称号，称号也可以空缺。
5. 用户点击完成按钮，该读书计划即可添加到用户的读书计划中。

备选流：

5a.用户点击取消按钮，该读书计划的创建即被取消。

## 修改读书计划

基本流：

1. 用户进入读书计划列表中。
2. 用户点击编辑按钮。
3. 用户进入某一个读书计划中修改。
4. 用户点击保存按钮。
5. 系统保存用户修改后的读书计划。

备选流：

4b.用户点击取消按钮，则已做的修改不保存，用例结束。

## 提出修改请求

基本流：

1. 用户进入读书计划列表中。
2. 用户选取一个已由自己修改过的读书计划。
3. 用户点击提出修改请求按钮。
4. 用户点击确定按钮。
5. 系统保存用户提出的修改请求。

备选流：

4b.用户点击取消按钮，则已做的修改不保存，用例结束。

## 获得成就

基本流：

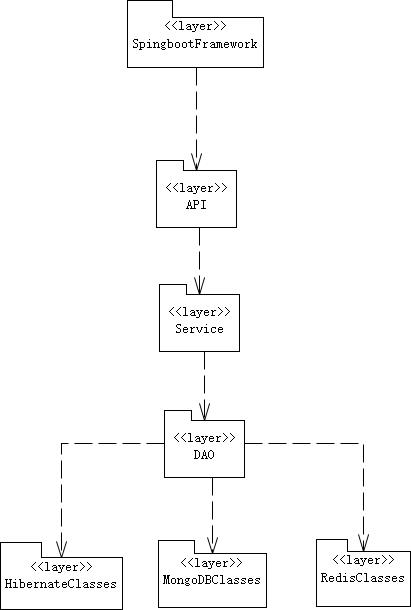
1. 用户进入某本书阅读进度中。
2. 用户点击编辑按钮。
3. 用户输入最新阅读进度。
4. 若达到解锁成就的要求，系统将自动为用户添加成就。

备选流：

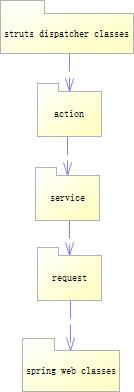
4b.用户未达到解锁成就的要求，用例结束。

# 逻辑视图

## 概述



如图为本项目的逻辑视图，主要分为四层，即controller层、service层、DAO层、middleware层。其中middleware层是对Hibernate、Morphia、Jedis相关类库的封装，用于对MySQL、MongoDB、Redis进行操作；DAO层依赖于middleware层，向上提供对数据库中的数据的增删改查操作；service中包含了相应的业务逻辑，通过调用DAO层的基本操作来实现。Controller层接收来自APP端和Web端的请求，并通过调用service层完成处理后把处理结果以JSON的形式返回给APP与Web端。



1. Struts dispatcher filter classes 响应页面请求，创建相应的action实例对请求进行处理。

2. action对象创建所需要的service实例，调用service的方法

3. service对象执行相应的功能，创建DodoAPIRequest抽象类的具体子类，创建org.springframework.web.client.RestTemplate实例

4. RestTemplate对象向dodo core api服务器发送http请求，并接收结果

## 在构架方面具有重要意义的设计包

### API

API包中包含了所有REST API的实现。APP与Web端可以通过访问URL来调用相应的API，实现相关操作，API包中每个Class通过调用service层的业务逻辑处理一个部分的操作（如对书的所有操作，增删改查等等）。

### Service

Service包中包含了所有的业务逻辑，如列出所有用户，查找用户想要的书等等，service包中的每个class通过调用DAO层对数据库的操作来实现一个模块的操作（比如搜索、罗列等等）。

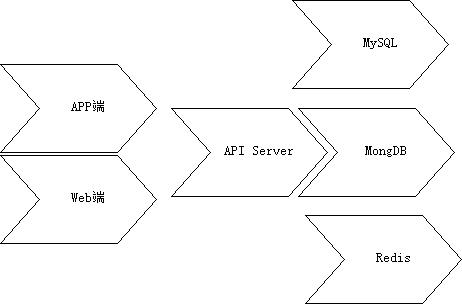
### DAO

DAO包中包含了所有对Hibernate、Morphia、Jedis等类库的操作以完成数据持久化操作，通过调用middleware来完成诸如增删改查的操作。

### Middleware

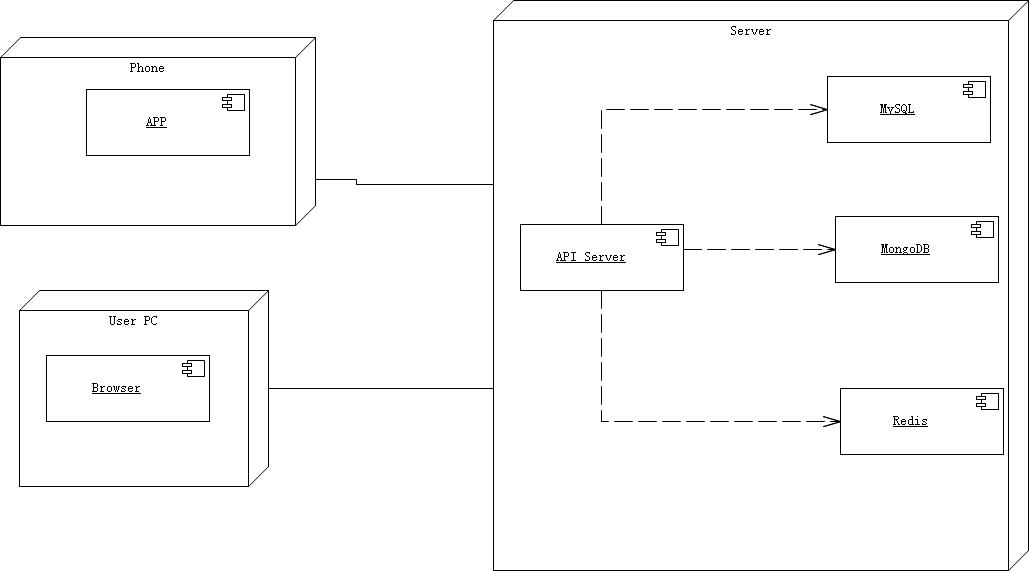
Middleware包中通过对Hibernate、Morphia、Jedis等类库的封装来完成数据持久化的工作。

# 进程视图



涉及到的主要进程如上图所示，APP与web server都通过REST API和API Server进行交互，API Server分别通过JDBC、MongoDB-java-driver、Jedis与MySQL、MongoDB、Redis三个进程交互。

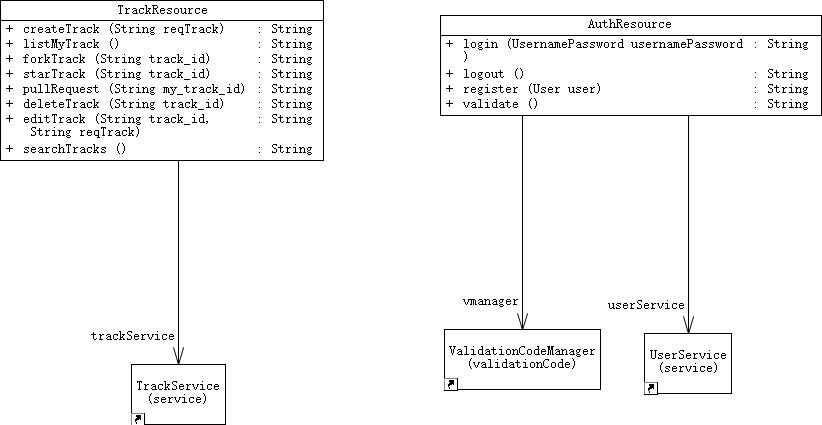
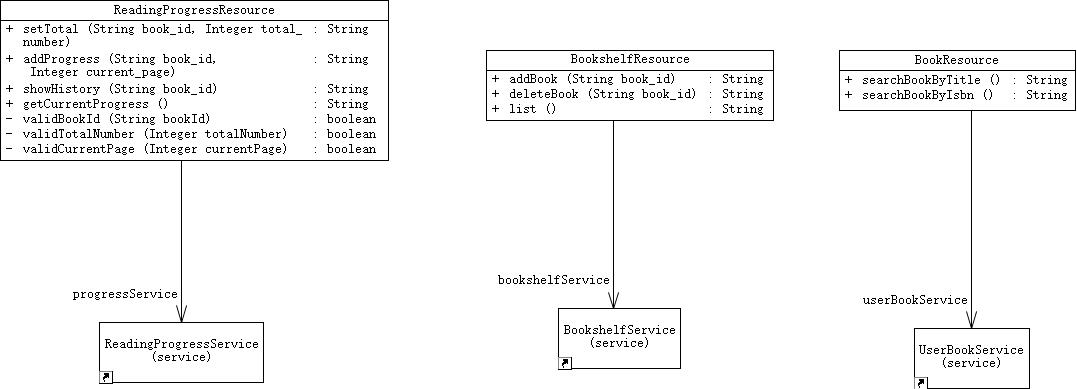
# 部署视图



所有的server及数据库进程全部部署于同一台云服务器上，APP端部署在iPhone上使用。

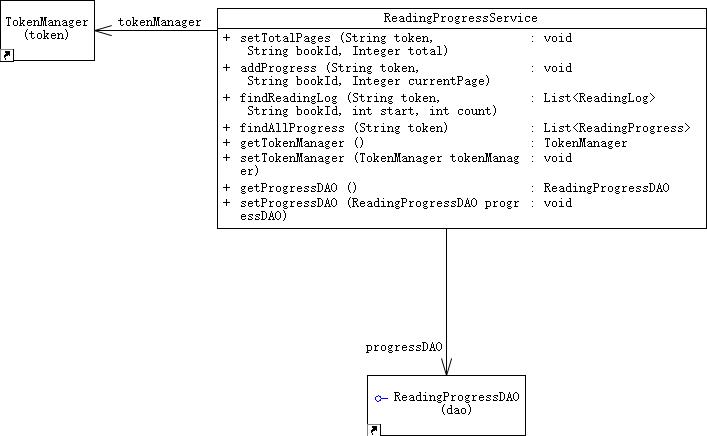
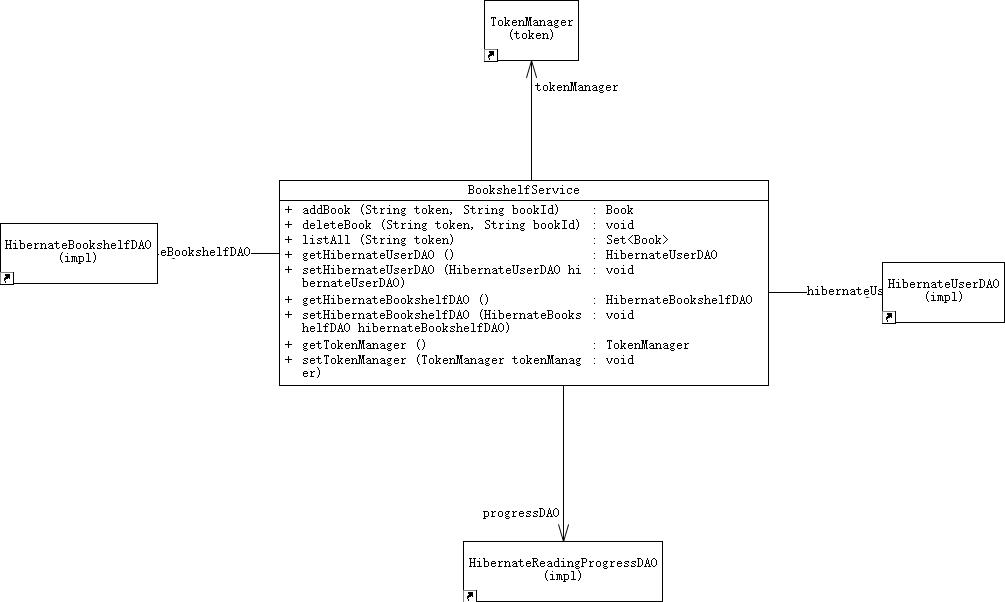
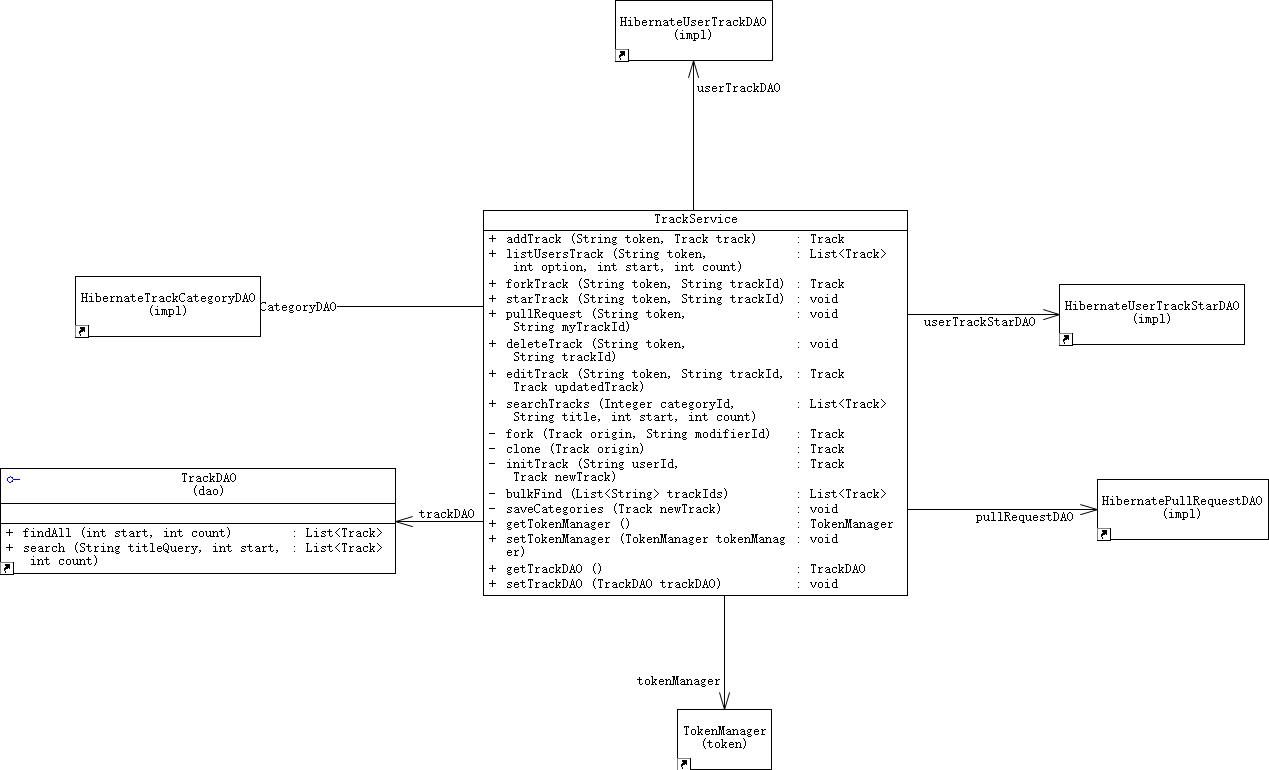
# 实现视图

## API



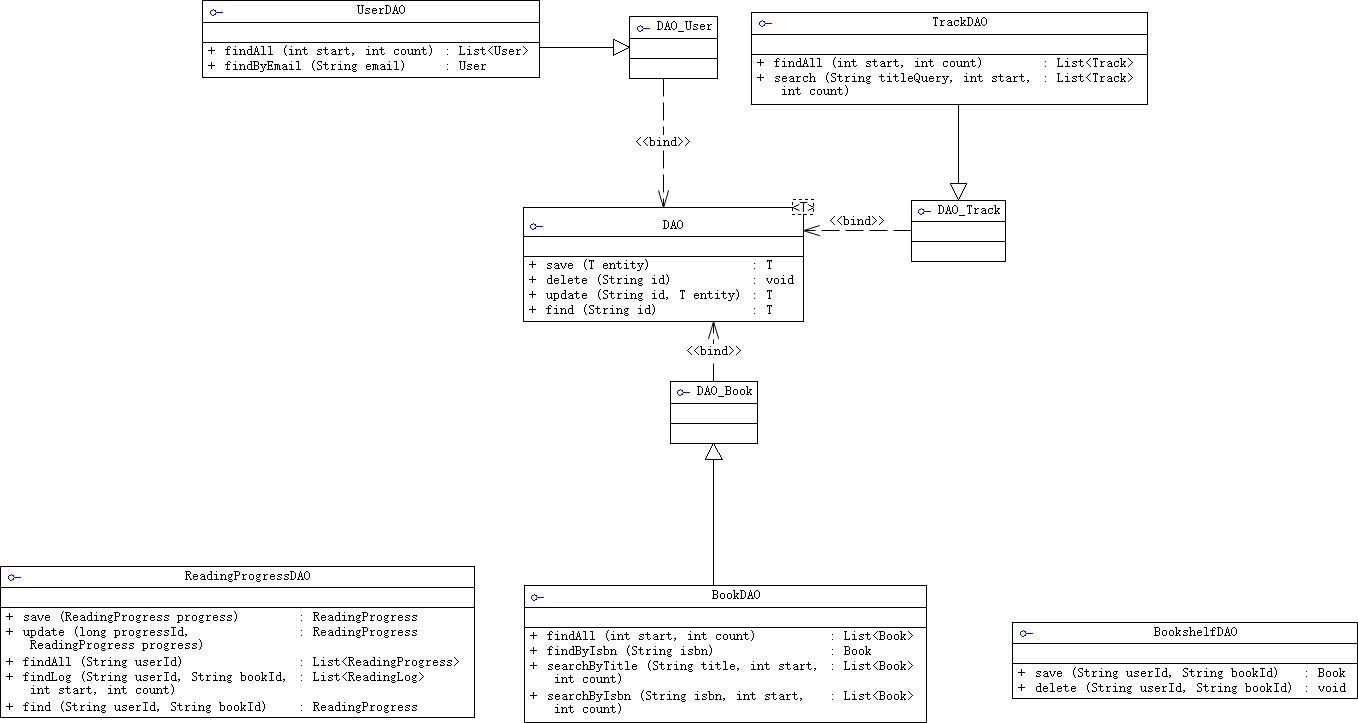
如上图所示，BookshelfResource处理所有与用户书架相关操作，BookResource处理所有与书籍本身相关操作，TrackResource处理所有与阅读计划相关的操作，AuthResource处理所有与权限管理相关的操作。

## Service



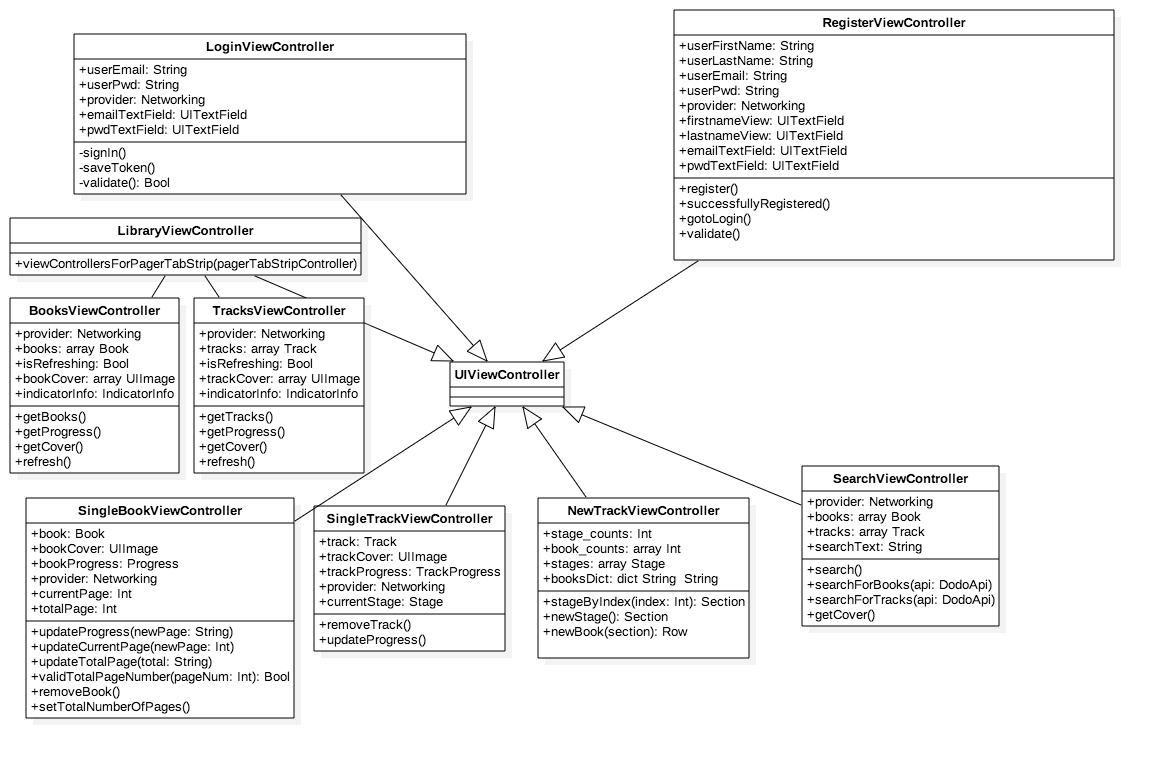
Service中的类设计思路与Resource相同，也是分别处理对书架、阅读进度、书籍本身的业务逻辑。

## DAO



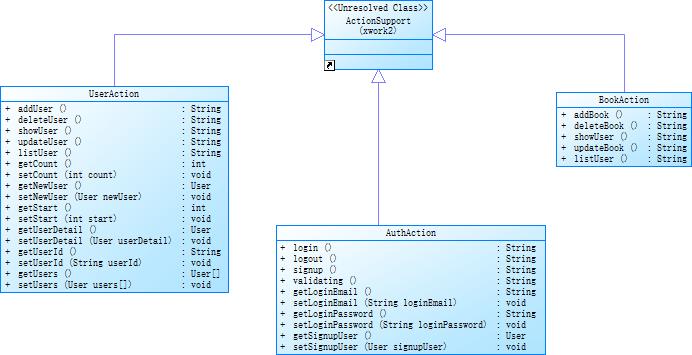
DAO为一个抽象接口，所有的UserDAO、TrackDAO、ReadingProgressDAO、BookDAO、BookshelfDAO都为其子类，分别完成基础的增删改查操作。

## APP

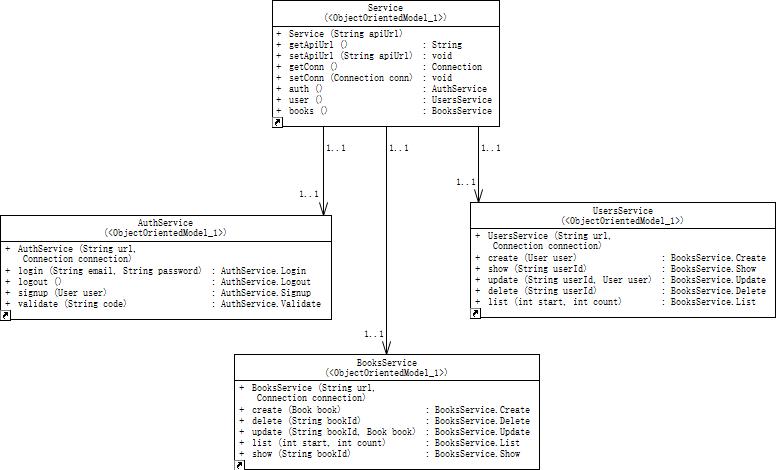


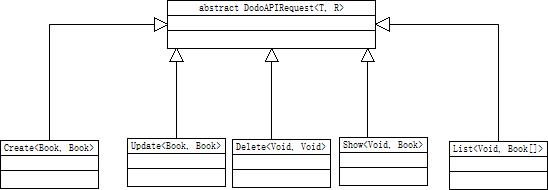
LoginViewController与RegisterViewController分别用于处理用户的登录、注册，而所有的与Library、Book、Track相关的ViewController完成与书架、书籍、阅读计划的管理。

## WebApp



AuthAction, UserAction, BookAction拓展com.opensymphony.xwork2.ActionSupport类，分别负责相应用户权限，user管理，书籍管理等请求



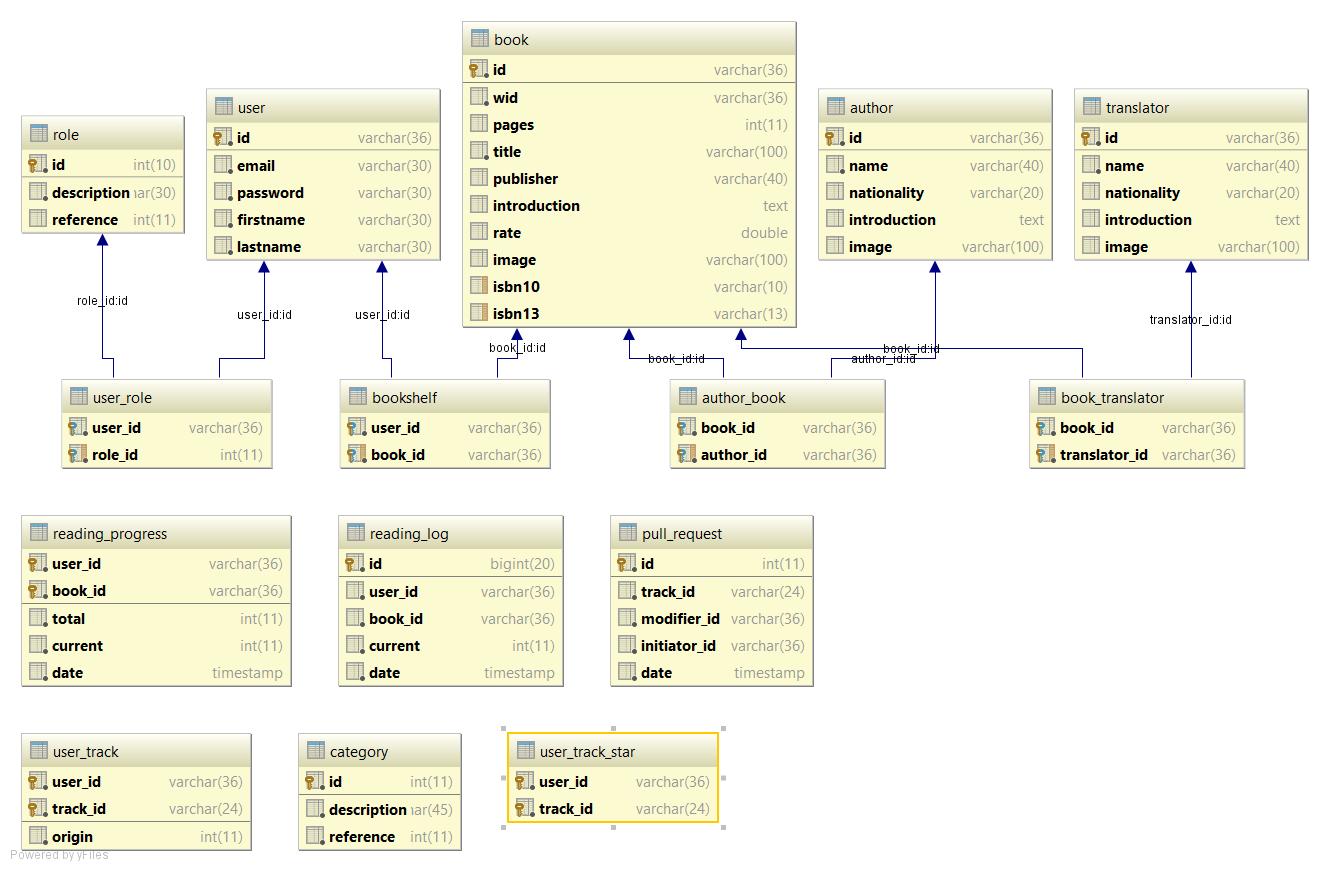


1. DodoAPIRequest抽象类负责发送和接收http请求，通过RESTful api与dodo核心服务器交互

2. 以CRUD book为例，create, update, delete, show, list都拓展了DodoAPIRequest<T, R>，T表示request body类型，R表示response body类型。在构造时设置url, header, body等信息

3. 调用父类request()方法，根据之前设置的url, header, body，构造org.springframework.web.client.RestTemplate并发出请求，设置返回结果

# 数据视图（可选）



# 核心算法设计（可选）