总体设计方案

# 设计思想

使用javaee架构mvc模式实现，前后端分离

表现层使用html+css+jquery+ajax

控制层使用springmvc

业务层使用service组件

持久层使用dao组件

# 技术架构

SSM框架：springmvc+spring+mybatis

开发环境：windows10和mac，tomcat、mysql、eclipse、axure（原型）、visio（er图）

采用技术：java+jquery+ajax+springmvc+IOC+AOP+mybatis

Java：核心开发语言

Jquery：简化前端js

Ajax：页面局部刷新，提升用户体验

Springmvc：接收请求、调用业务组件、生成json返回给前端

Spring：管理相关组件

IOC：管理controller、servlet、dao，维护各层关系

AOP：面向切面编程，不修改原有的代码，给系统增加新功能

# 系统架构

系统总体架构包含数据存储层、应用层、web负载层。采用前后端分离架构。

## 前端架构

web 浏览器

数据展示

HTML

CSS

AJAX

JQUERY

GIT

API接口层

JSON

HTTP

UEDIT

BOOTSTRAP

### UEditor 介绍

UEditor （[https://ueditor.baidu.com](https://ueditor.baidu.com/)）是由百度「FEX前端研发团队」开发的所见即所得富文本web编辑器，具有轻量，可定制，注重用户体验等特点，开源基于MIT协议，允许自由使用和修改代码。

## 后端架构

前端应用

控制层

SSM

SpringMVC

Spring

IOC

AOP

GIT

JDBC

JSON

MYSQL

数据持久化

MAVEN

Mybatis

JACKSON

JSON接口服务

# 开发规范

所有请求通过ajax访问

前端页面使用html

请求响应格式为json：{status:xxx,msg:xxx,data:xxx}

## URL规范

url全部小写

return View 全部驼峰大写

URL mapping

|  |  |  |
| --- | --- | --- |
| controller | URL | mapping |
| LogRegController | /my\_cloud\_note/ | Login.jsp |
| LogRegController | /my\_cloud\_note/login | Login.jsp |
| LogRegController | /my\_cloud\_note/regist | Regist.jsp |
| UserController | /my\_cloud\_note/user/login.do | Edit.jsp  Login.jsp |
|  | /my\_cloud\_note/user/regist.do | Login.jsp  Regist.jsp |

# 代码包划分

com.wlfei.mcn.dao 接口层

com.wlfei.mcn.service 服务层

com.wlfei.mcn.controller 控制层

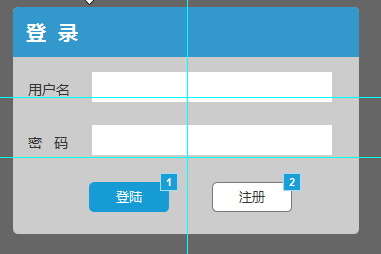
com.wlfei.mcn.entity 实体层

com.wlfei.mcn.util 工具层

com.wlfei.mcn.aspect 切面层

# 功能实现逻辑

## 登录功能



### 发送ajax请求

输入用户名、密码，

点击登录按钮，触发绑定事件

事件获取参数，

事件发送ajax请求

### 服务器处理

请求

-->DispathcerServlet-->HandlerMapping-->Controller.execute()

-->Service(

//检查用户和密码

//错误则返回错误json，否则返回登陆成功json

)

-->Dao-->Mysql(cn\_user)-->返回JSON数据

### ajax回调处理

登陆成功，进入edit.html

登陆失败，提示用户名或密码错误，重新登陆。

## 注册功能



### 发送ajax请求

1. 点击注册按钮，触发绑定事件
2. 事件获取参数
3. 事件发送ajax请求

### 服务器处理

请求

-->DispathcerServlet-->HandlerMapping-->Controller.execute()

-->Service(

//检查用户

//存在重复用户则返回json，否则添加用户

//处理密码

//保存用户密码

)

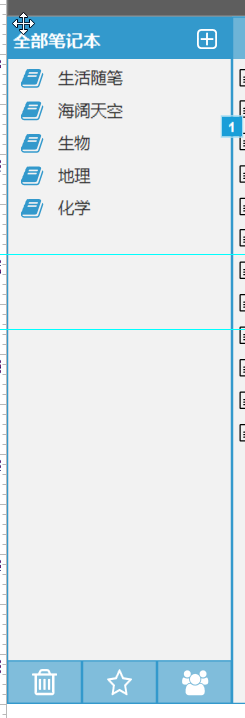
-->Dao-->Mysql(cn\_user)-->返回JSON数据

### ajax回调处理

成功：注册成功

失败：用户名已存在

## 笔记本（book）列表显示功能



### 发送ajax请求

1. 进入edit.html页面时请求
2. 请求参数：userId
3. 发送json

### 服务器处理

请求

-->DispathcerServlet-->HandlerMapping-->Controller.execute()

-->Service(

//笔记本service接口--查找用户笔记本

//实现类调用dao去查找笔记本

//实现类构建result

//笔记本dao根据userId查找

//mysql表：cn\_notebook

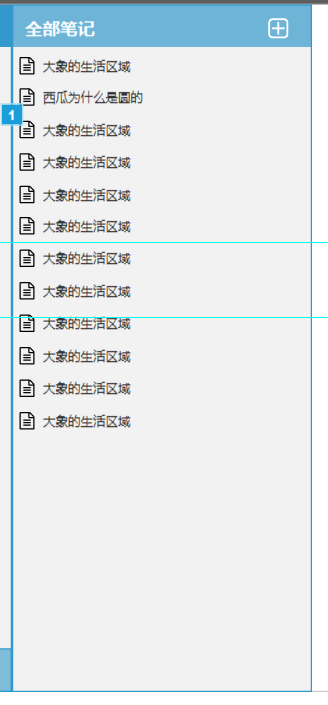
)

-->Dao-->Mysql-->返回JSON数据

### ajax回调处理

解析json，循环生成笔记本列表项

## 笔记（note）列表显示功能



### 发送ajax请求

1. 点击笔记本li
2. 请求参数为笔记本的id--bookId
3. 发送json请求到服务器

### 服务器处理

请求

-->DispathcerServlet-->HandlerMapping-->Controller.execute()

-->Service(

//笔记service接口--加载笔记

//实现类调用dao去查找笔记本

//实现类构建result

//笔记本dao根据bookId查找

//mysql表：cn\_note&cn\_note\_status

)

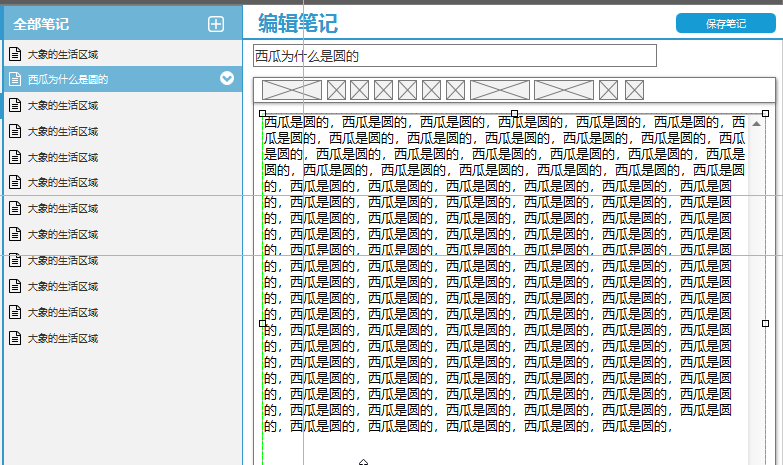
-->Dao-->Mysql-->返回JSON数据

### ajax回调处理

成功：解析json，生成笔记list，添加到笔记列表中，注意清楚原来的列表

失败：提示笔记加载失败

## 显示笔记内容



### 发送ajax请求

1. 笔记列表li单击
2. 请求参数：noteId
3. 发送ajax请求

### 服务器处理

请求

-->DispathcerServlet-->HandlerMapping-->Controller.execute()

-->Service(

//笔记service接口--加载笔记

//实现类调用dao去查找笔记

//实现类构建result

//笔记本dao根据noteId查找

//mysql表：cn\_note)

-->Dao-->Mysql-->返回JSON数据

### ajax回调处理

成功：

1. 设置笔记标题
2. 设置笔记内容

重点是使用了uedit前端插件

失败：笔记信息加载失败

## 更新笔记或保存笔记

### 发送ajax请求

1. 点击保存笔记按钮
2. 请求参数：noteId、title、body
3. 发送ajax请求

### 服务器处理

请求

-->DispathcerServlet-->HandlerMapping-->Controller.execute()

-->Service(

//笔记service接口--更新笔记

//实现类调用dao去更新笔记

//实现类构建result

//笔记本dao更新note

//mysql表：cn\_note

)

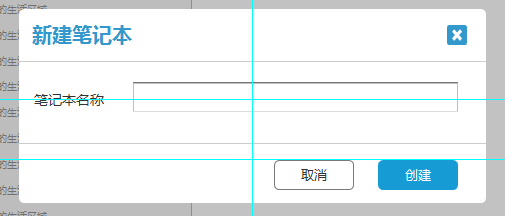
-->Dao-->Mysql-->返回JSON数据

### ajax回调处理

成功：对被更新的笔记名称进行标记

失败：提示更新失败

## 创建一个新的笔记本



### 发送ajax请求

1. 单击创建按钮
2. 请求参数：用户ID、笔记本名称
3. 发送ajax请求

### 服务器处理

请求

-->DispathcerServlet-->HandlerMapping-->Controller.execute()

-->Service(

//笔记本service接口--新增笔记本

//实现类调用dao去创建笔记本

//实现类构建result

//笔记本dao根据Book实例保存

//mysql表：cn\_notebook

)

-->Dao-->Mysql-->返回JSON数据

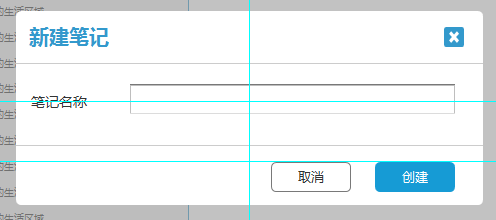
### ajax回调处理

成功：关闭对话框、添加一个笔记本li

提示创建笔记本成功

失败：提示创建笔记本失败

## 创建一个新的笔记



### 发送ajax请求

1. 点击笔记处的新建按钮
2. 请求参数：笔记名称、笔记本ID、用户ID
3. 发送ajax请求

### 服务器处理

请求

-->DispathcerServlet-->HandlerMapping-->Controller.execute()

-->Service(

//笔记service接口--添加笔记

//实现类调用dao去添加笔记本

//实现类构建result

//笔记本dao添加笔记

//mysql表：cn\_note

)

-->Dao-->Mysql-->返回JSON数据

### ajax回调处理

成功：关闭对话框、解析json生成li到笔记本列表（title、noteId）、提示创建成功

失败：提示创建失败

## 显示笔记本下拉菜单



点击箭头，显示三个菜单项

### 分享功能

#### 发送ajax请求

1. 点击分享
2. 请求参数：noteId
3. 发送ajax请求

#### 服务器处理

创建share类

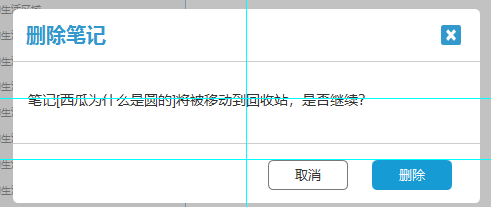
cn\_share表新增一条数据

#### ajax回调处理

成功：提示分享成功，笔记添加一个分享图标

失败：提示分享失败

### 删除功能



#### 发送ajax请求

1. 点击删除按钮
2. 请求参数：noteId
3. 发送ajax请求

#### 服务器处理

noteService删除（软删除）

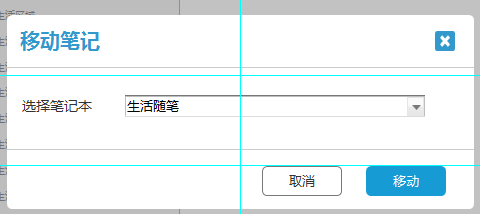
dao更新笔记状态为已删除

#### ajax回调处理

成功：提示删除成功，获取笔记id，列表中删除笔记

失败：提示删除失败

### 移动功能



#### 发送ajax请求

1. 页面获取所有笔记本名称和id，删除现在所在的笔记本
2. 创建选择对话框
3. 选择笔记本
4. 请求参数：笔记ID、笔记本ID
5. 发送ajax请求

#### 服务器处理

更新笔记本：笔记对应关系

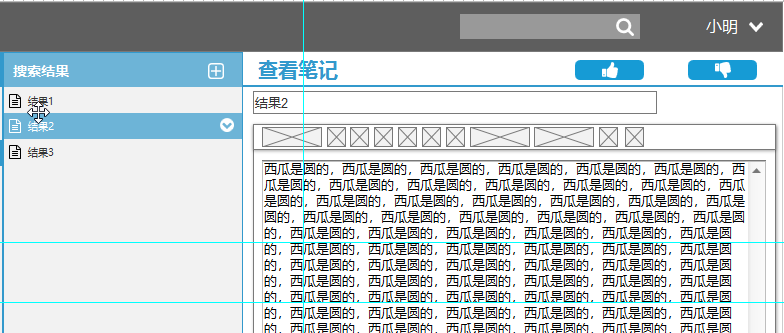
返回json

#### ajax回调处理

成功：当前列表删除笔记

失败：提示移动失败

## 模糊查询分享笔记



### 发送ajax请求

1. 顶部的搜索栏输入关键字，按回车
2. 请求参数：关键字
3. 发送ajax请求

### 服务器处理

查询分享中的笔记，并返回

### ajax回调处理

成功：

笔记栏标题修改为：搜索结果

笔记栏清空

笔记栏写入新的li

失败：提示搜索失败

## 修改密码和退出下拉



## 修改密码



### 发送ajax请求

1. 弹出修改密码对话框
2. 输入原密码、新密码
3. 点击修改
4. 请求参数：原密码、新密码、确认密码
5. 发送ajax请求

### 服务器处理

更新用户密码

返回json

### ajax回调处理

成功：提示修改密码成功

失败：提示修改失败

## 退出登录

删除session

跳转到登陆页面

## AOP应用

AOP：在软件业，AOP为Aspect Oriented Programming的缩写，意为：面向切面编程，通过预编译方式和运行期动态代理实现程序功能的统一维护的一种技术。AOP是OOP的延续，是软件开发中的一个热点，也是Spring框架中的一个重要内容，是函数式编程的一种衍生范型。利用AOP可以对业务逻辑的各个部分进行隔离，从而使得业务逻辑各部分之间的耦合度降低，提高程序的可重用性，同时提高了开发的效率。

性能审计：查看各个业务执行时间

异常保存：异常信息保存到日志

日志：封装日志操作

# 应用部署架构

nginx

tomcat

mysql

web层

应用层

数据层