目 录

[第一章 项目介绍 1](#_Toc110449178)

[1.1文档编制目的 1](#_Toc110449179)

[1.2项目开发背景 1](#_Toc110449180)

[1.3项目特点 1](#_Toc110449181)

[1.4项目开发环境 2](#_Toc110449182)

[第二章 项目总体结构 2](#_Toc110449183)

[2.1源码目录介绍 3](#_Toc110449184)

[2.2数据库表 7](#_Toc110449185)

[2.3模块大体功能简介 8](#_Toc110449186)

[第三章 模块需求 8](#_Toc110449187)

[3.2. 公共功能 9](#_Toc110449188)

[3.3登录 10](#_Toc110449189)

[3.3.1.1用户登录 10](#_Toc110449190)

[3.4. 系统主页 11](#_Toc110449191)

[3.5. 系统管理员功能 11](#_Toc110449192)

[3.5.1. 用户管理 11](#_Toc110449193)

[3.5.2 权限管理 13](#_Toc110449194)

[3.5.4 文章管理 13](#_Toc110449195)

[3.6. 普通用户功能 14](#_Toc110449196)

[3.6.1 文章浏览 14](#_Toc110449197)

[3.6.1 文章操作 15](#_Toc110449198)

[第四章 项目总体评价 16](#_Toc110449199)

[4.1. 用户界面评价 16](#_Toc110449200)

[4.1.1. 用户界面设计基本原则 17](#_Toc110449201)

[4.1.2. 用户界面设计规范 17](#_Toc110449202)

[4.2. 功能性评价 18](#_Toc110449203)

[第五章软件设计 19](#_Toc110449204)

# 第一章 项目介绍

博客就是开放大个人空间，可以充分利用超文本链接、网络互动、动态更新等特点，在网络中，精选并链接全球互联网中最有价值的信息知识与资源；也可以将个人工作过程、生活故事、思想历程、闪现的灵感等及时记录和发布，发挥个人无限的表达力；更可以以文会友，知识和汇聚朋友,进行深度交流和沟通。文章通过博客作为载体，将文章不断的传播，让人不断的思考，不断的进步，永不停歇的探索宇宙的尽头。

## 1.1文档编制目的

编写此文档的主要目的是明确所要开发的软件所应具有的功能，使系统分析人员和软件设计人员能清楚地了解用户的需求，并在此基础上进一步提出概要设计和完成后续设计与开发工作，为软件开发范围、业务处理规范提供依据，也是应用软件进行最终验收的依据。

## 1.2项目开发背景

博客的开发精神是：自由表达、开放宽容、专业精神。

在没有自己的博客之前，人们会经常进出于论坛发表帖子或者通过即时通信软件聊天，来表达自己的想法，可是这些都是零散的和杂乱的。博客的出现，让人们可以不断的把自己以前的和每天激发的一些想法和思考写入博客，在博客中交流寻找到自己的伯乐。写博客也是在自己的每一天留下一道浅浅的思考和划痕。

## 1.3项目特点

此系统具有如下3个特点:

➢灵活性：本系统可以根据不同登录用户，自动识别用户身份，并引导和呈现出该用 户可以进行的操作，同时系统管理员可以灵活的配置用户的具体操作权限。

➢简单便捷的用户操作: 功能模块的操作采用简单易行的使用原则，用户可以很容易理解某个操作的含义并很容易上手。

➢友好的用户界面:系统的操作界面简单、 美观、大方，能够给用户一种简洁舒适的感觉。

## 1.4项目开发环境

●设备配置

硬件平台:英特尔I3处理器，4G内存，80G 硬盘空间。

软件平台: Visual Studio Code，数据库Linux服务器

●工具及语言

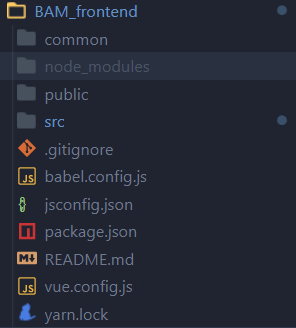
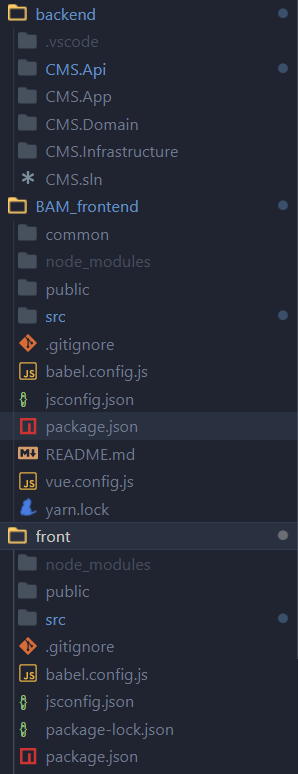
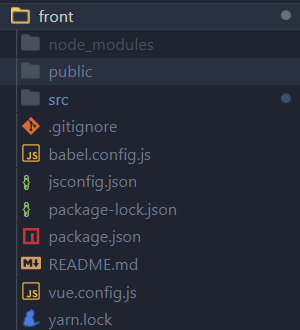
前端：AxureRp、ASP.Net Core vue/cli、AjAx、EF Core

后端：ASP.Net Core Webapi、Postgresql

# 第二章 项目总体结构

一个设计良好的项目架构必备条件是：易维护性、可扩展性、当遇到需求变更或功能变更时能以最低的代码成本响应变更。这就要求整个软件在设计上做好周密、全方位的设计。

此软件总体设计如下。

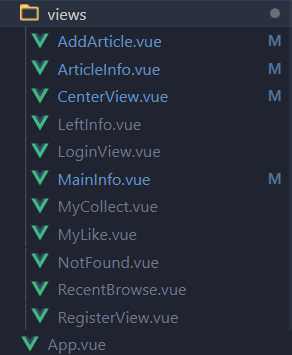
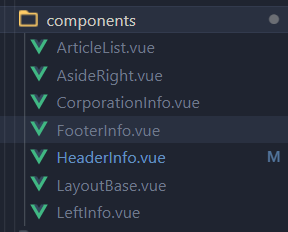
## 2.1源码目录介绍

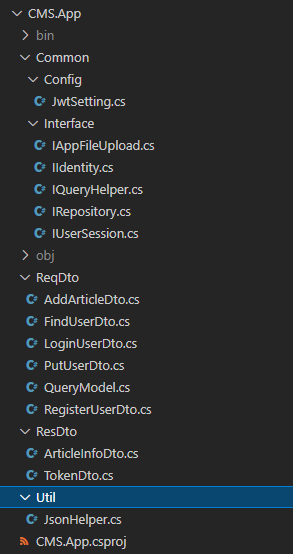
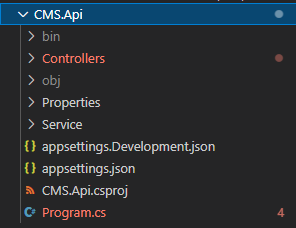
我们现在从整个项目的源码结构上做如下分析，以下是对项目源码部分进行分开介绍，如下：

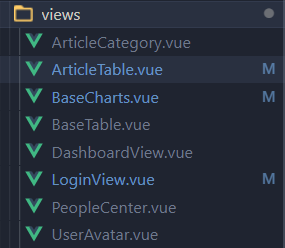
各项目层次说明如下

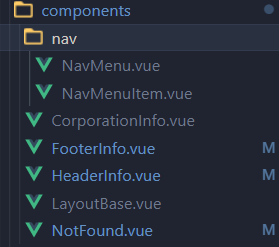
|  |  |
| --- | --- |
| 项目 | 说明 |
| CMS.Api | 用于对外提供接口和服务 |
| CMS.App | 主要逻辑需要的接口和业务 |
| CMS.Domain | 领域实体，只是提供实体 |
| CMS.Infrastructure | 基础设施层，所有需要利用外部工具的地方，都实现在这里如EFCore等，后续也方便换成别的Orm工具，而不会影响到其它的项目和逻辑 |
| views | 用于展示前端视图 |
| components | Vue组件 |

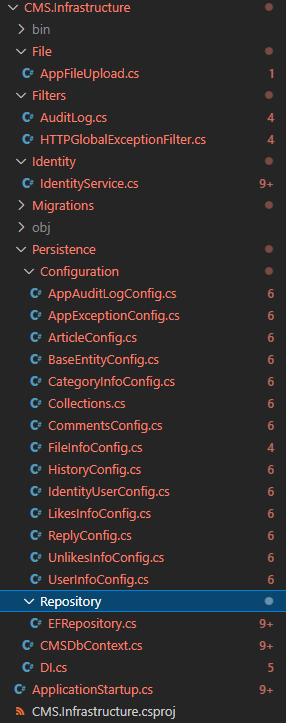
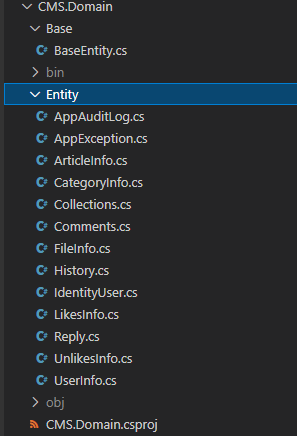
各层源码展开如下图所示：

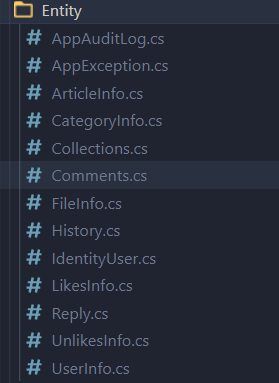




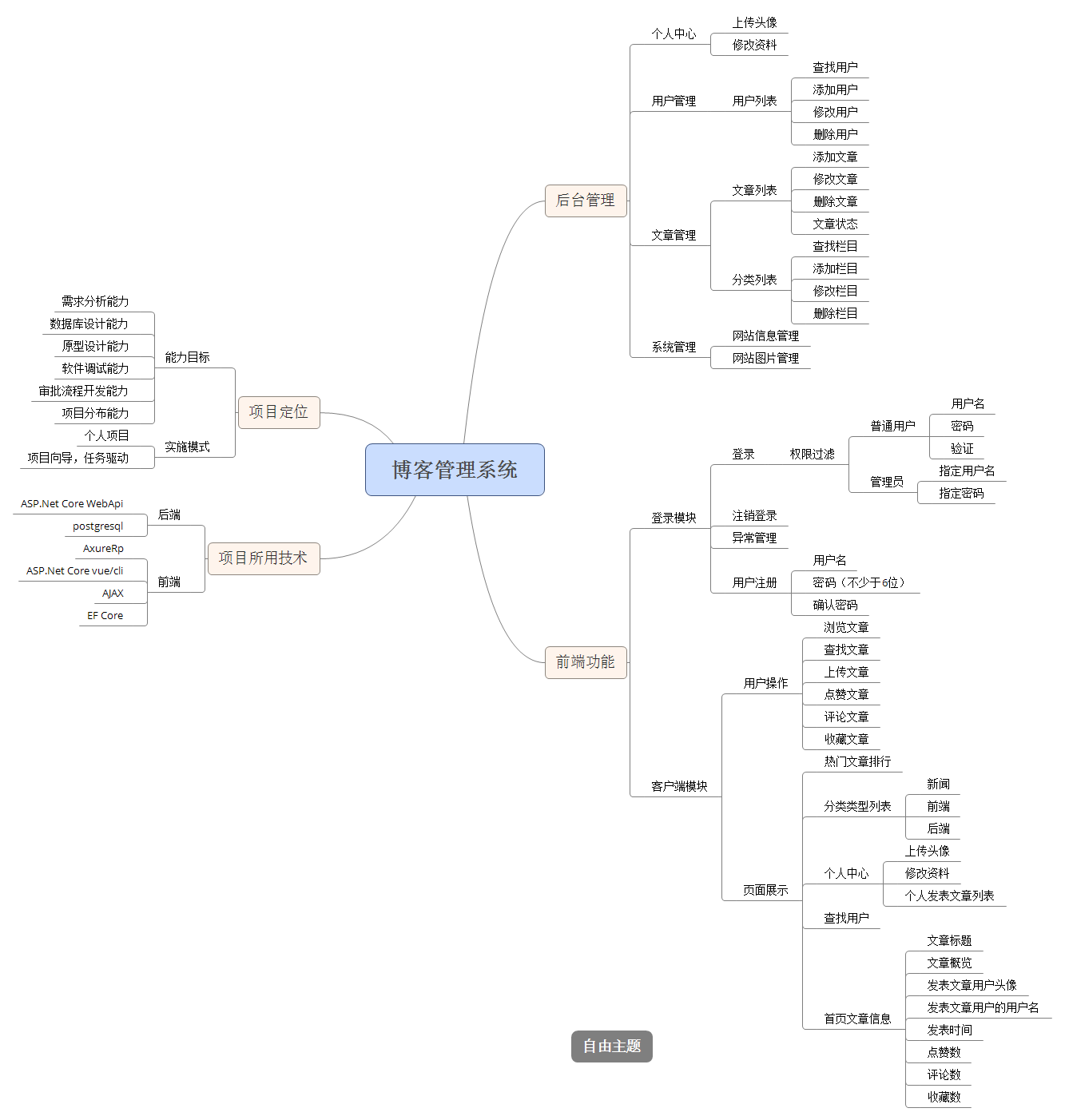




## 2.2数据库表



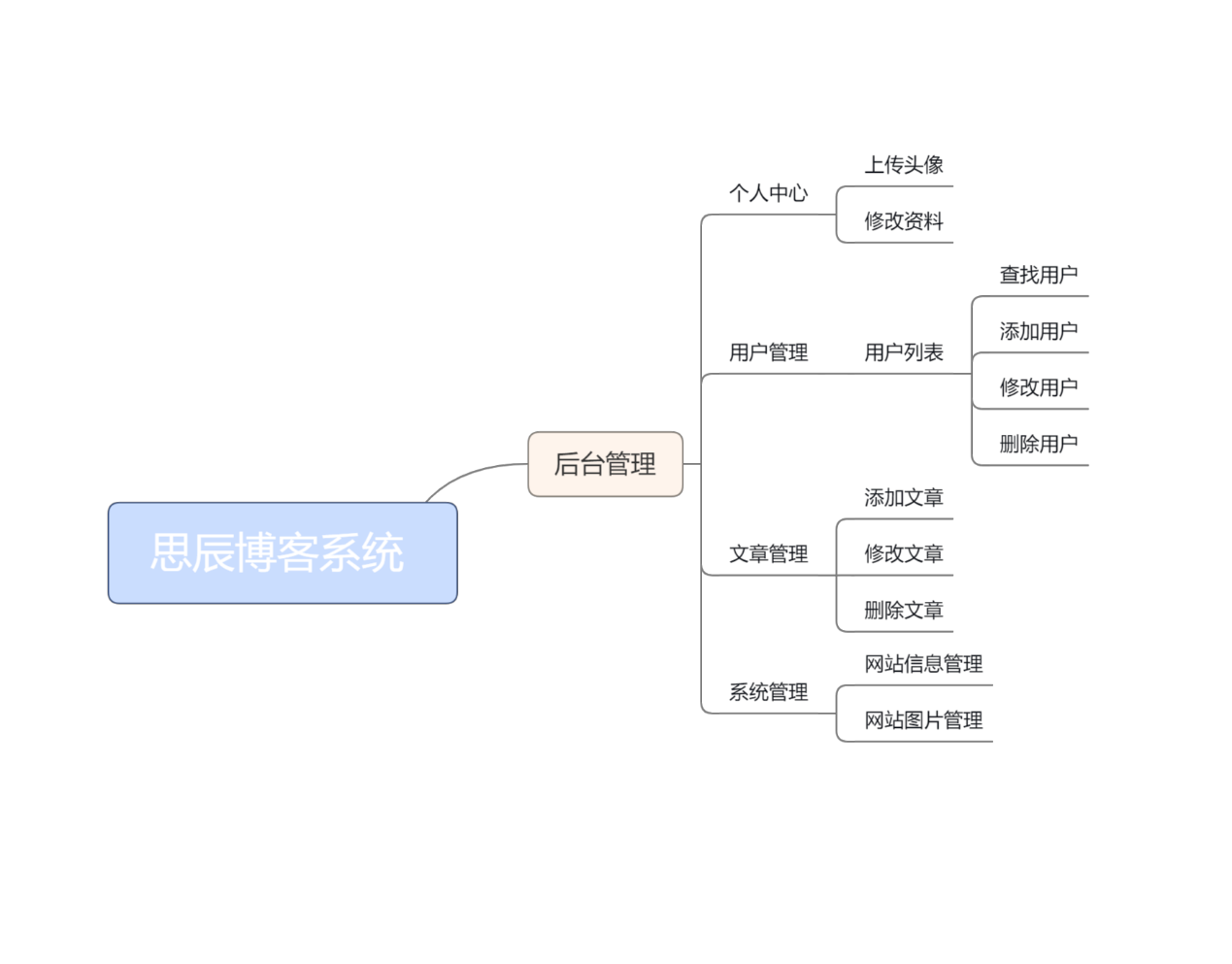
## 2.3模块大体功能简介



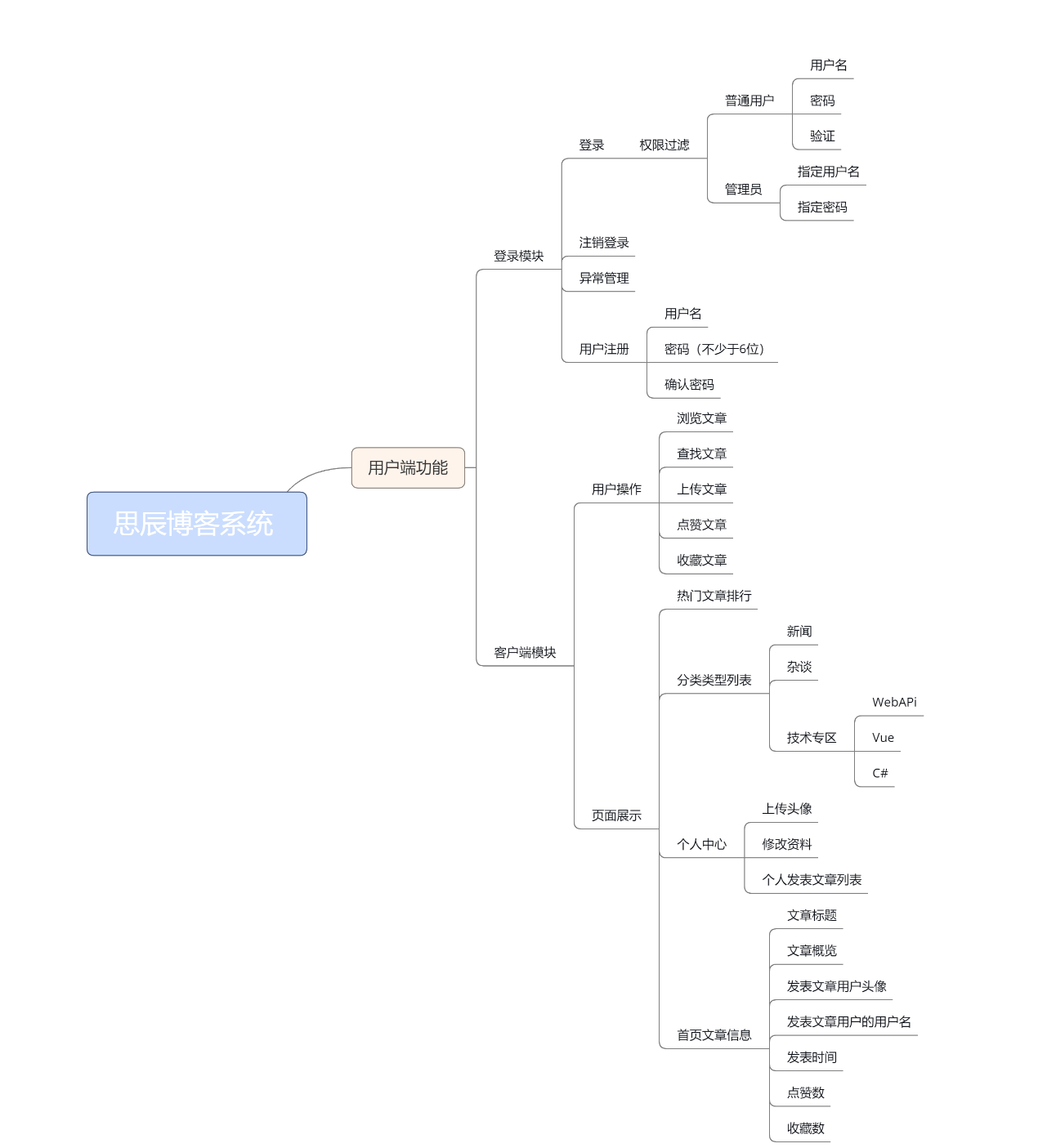
第三章 模块需求3.1.系统基本功能角色分工

一个好的应用界面的必备条件是：内容清楚、指示明白、屏幕美观和有亲切感。界面通常包含图形和文字。应用界面的设计是对控件进行适当的舍取及功能的选择和处理过程。在程序设计中需要对设计的方法反复推敲才能使其达到尽可能完美。

本系统基本角色有两种系统管理员，用户角色，进入系统后，根据权限可以操作的模块如下（系统管理员也可以根据需求的变更灵活的配置系统角色和权限）



3.2. 公共功能公共功能为使用本系统所有用户都具有的功能。其中包括登录和进入主页以及密码修改。



## 3.3登录

### 3.3.1.1用户登录

➢信息来源：所有用户

➢信息要求：用户登录系统及其身份验证

➢事件信息系统处理：只有用户填写正确的账号、密码、验证码才能登录系统并进入系统主页，否则会有相应的提示：例如“验证码不正确”或者“账号/密码错误”。



3.4. 系统主页 3.4.1. 系统主页

信息来源:所有的用户  
信息要求:不同的角色用户展示不同的目录

例如:用户登录展示用户界面，可进行相应的操作，管理员登录进入后台管理系统，对不同的系统管理进行操作  
事件信息系统处理：普通用户可进行登录注册，不符合逻辑要求将提示报错，可对文章进行浏览、点赞、评论等。管理员可对文章、用户和系统设置进行查看，添加，修改，删除等操作。相应的功能对应相应的信息。

## 3.5. 系统管理员功能

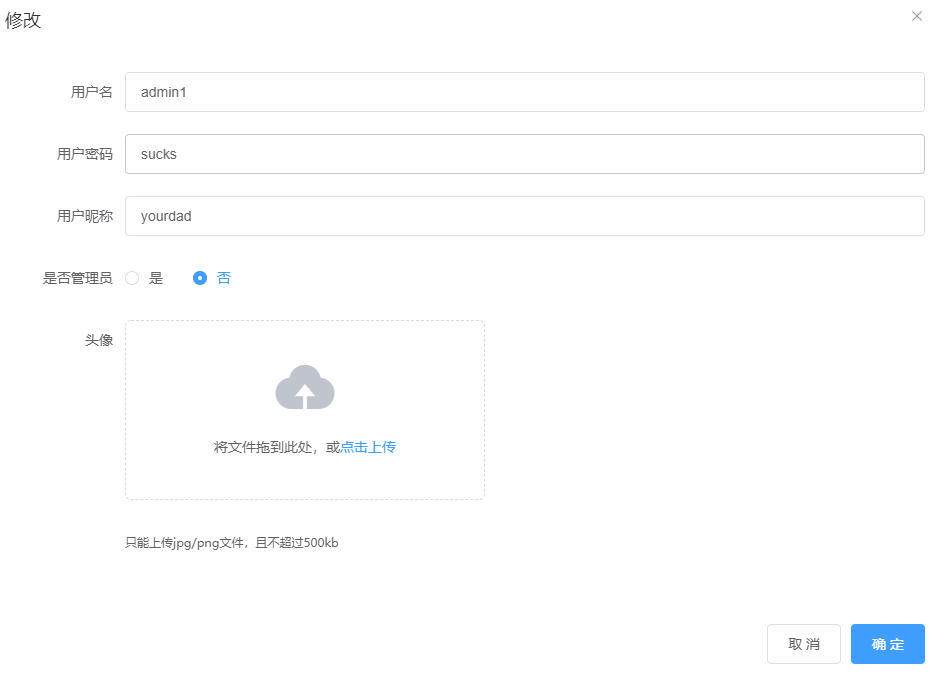
3.5.1. 用户管理信息来源：系统管理员  
 信息要求：展现所有用户的注册信息并且可对其进行相应的操作，并具有回收站功能

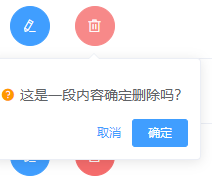
事件信息系统处理：

1. 在文本框输入用户名，点击搜索按钮，可以模糊查询到用户的信息，并展示处理

2.点击添加用户，弹出相应信息的窗体页面进行添加（如图）



3.点击修改用户，弹出相应信息的窗体页面进行修改，且弹窗页面在未修改前的文本默认为原用户信息（如图）  


4.点击删除用户弹出相应的弹窗页面，点击确定将删除该用户（如图）  


3.5.2 权限管理信息来源：系统管理员  
 信息要求：对系统权限的设置   
 事件信息系统处理：

1. 在文本框输入用户名，点击搜索按钮，可以模糊查询到用户的信息，并展示处理（如图）  
   

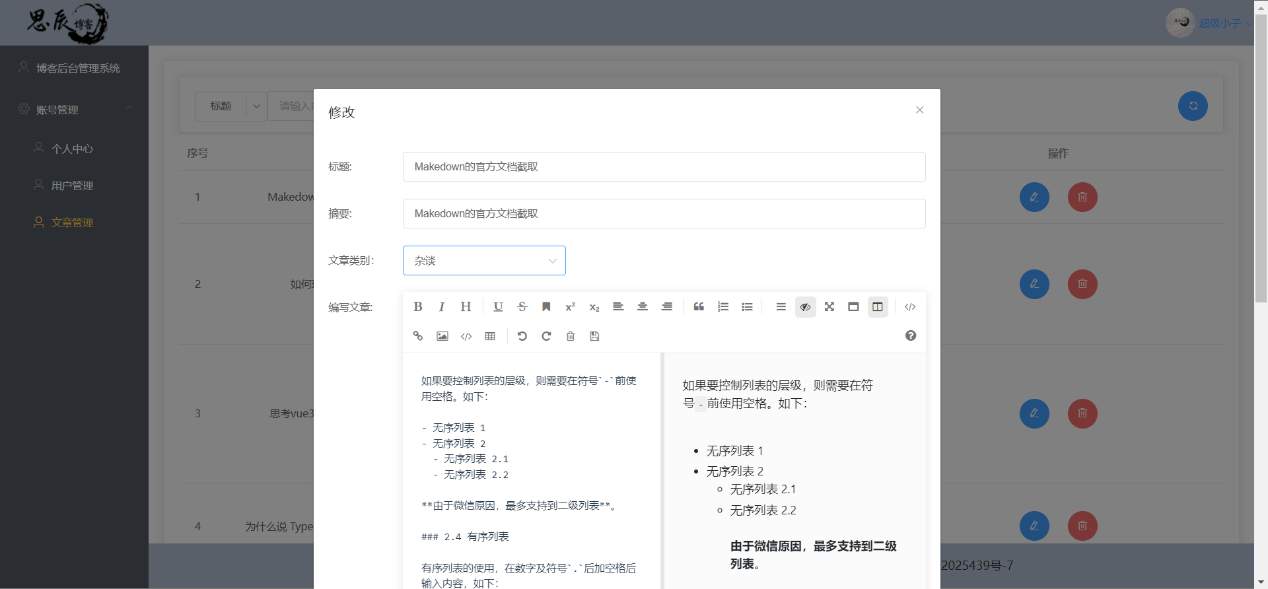
3.5.4 文章管理  
信息来源：系统管理员  
 信息要求：对文章的增加、删除、修改和查询，与回收站的恢复删除功能  
 事件信息处理：（如图）|  
1.查：在文本输入框输入文章信息，对文章进行模糊查询(如图)



1. 删：点击删除按钮，弹出提示框（如图），点击确定，则将文章删除(如图)



3.改：点击修改图标，弹出响应的弹窗页面(如图)



## 3.6. 普通用户功能

### 3.6.1 文章浏览

信息来源：普通用户  
 信息要求：在用户端主页面可以实现文章浏览   
 事件信息处理：（如图）|  




### 3.6.1 文章操作

信息来源：普通用户  
 信息要求：在用户端主页面可以实现添加文章，文章点赞，点踩，收藏   
 事件信息处理：（如图）|

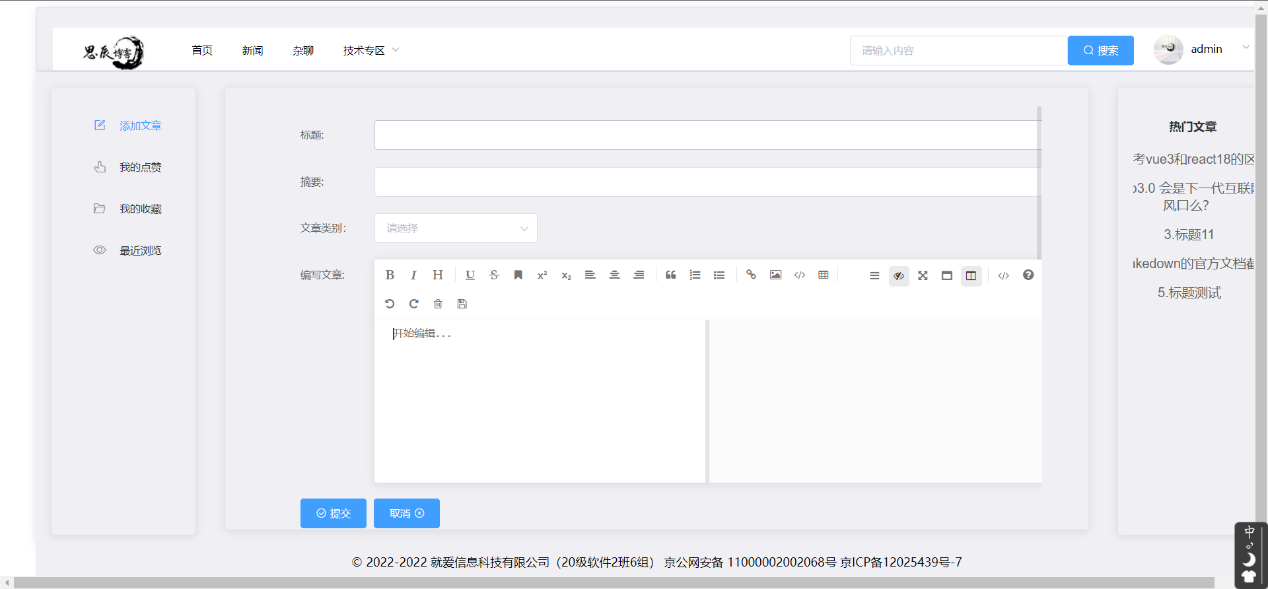
1.点击点赞图标，点赞数量加一，并且在“我的点赞”页面可以看到最近点赞过的文章。为了防止人为刷赞，对点赞按钮添加验证，每个用户只能对一篇文章点赞一次，点踩过的文章不能进行点赞操作。



2.点击收藏图标，收藏数量加一，并且在“我的收藏”页面可以看到最近收藏过的文章。



1. 添加文章，可以在添加文章页面编写文章的标题、摘要、分类和正文。正文部分支持Makedown格式。



# 第四章 项目总体评价

## 4.1. 用户界面评价

用户界面交互简单，UI设计精简，一切以用户体验为主。减少用户的视觉疲劳和学习适应成本。

### 4.1.1. 用户界面设计基本原则

1.使用一致的术语、一致的步骤、一致的动作，让用户始终用同一种方式思考与操作。最忌讳每换一个屏幕就更换一套操作命令与操作方法。

2.富有吸引力 用户界面不仅应该方便、高效,还应该使用户在使用时感到兴奋,能够从中获得乐趣，从而吸引用户更乐意使用它、接受它。

3.界面元素的位置 用户界面是由若干界面元素组成的，不是所有元素都一样重要，要确保重要的元素尽快显现给用户。重要的或频繁访问的元素应当放在显著的位置上，而不太重要的或使用频率小的放到不太显著的位置上。

4.保持界面元素的一致性 一致的外观可以在系统中创造一种和谐。如果界面元素缺乏一致性，很可能使界面看起来杂乱无章、没有条理，甚至可能引起用户对系统可靠性的怀疑，失去对系统应用的信心。5.合理使用空白区域 当然此处的“白”并非指白色，而是指界面中没有被界面元素占用的空白区域。恰当地“留白”有助于突出界面元素和改善可用性，引导用户在不知不觉中进入系统的思维框架，从而顺利地、正确地应用这个系统。

5.慎重使用颜色 在界面上合理地使用颜色可以增加视觉上的感染力，但滥用颜色会适得其反。一般而言，亮色容易使人兴奋，同时也容易使人的眼睛产生疲劳；柔和的暗色可以使人平静。

6.字体的使用 文字是传达信息的工具，有文字就有字体。因此字体也是影响用户界面质量的重要因素。程序员要选取在不同的分辨率和不同类型的屏幕上都能顺利阅读的字体。另外，字体的一致性也很重要，在大多数情况下，不应当采用两种以上的字体

7.按钮类元素 设计界面时仔细分析各种元素的功能以及元素之间的逻辑关系，并按功能或关系进行逻辑分组，使用控件组织起来，以加强元素之间的联系。另外，如果界面放置的元素太多，可考虑采用菜单来完成相同的功能。

**8.列表框类元素**列表框类元素有一个共同的特点，都可以设置滚动条。使在一个小“框”中阅读大文件成为可能。但也产生一个问题：如果阅读一个行列均很多的文件，用户就会反复拉动水平或垂直滚动条，给阅读带来诸多不便。设计时最好只使用一个方向的滚动条，即要么横向够宽，要么纵向够长。

**9.控件隐藏**把当前状态下无用的元素隐藏起来，既能使界面清洁明了，突出重点，又能保证用户不做无效的“工作”。

4.1.2. 用户界面设计规范 **一般交互原则**      
  1）   一致性：菜单选择、数据显示以及其它功能都应使用一致的格式。    
  2）   提供有意义的反馈    
  3）   执行有较大破坏性的动作前要求确认    
  4）   在数据录入上允许取消大多数操作    
  5）   减少在动作间必须记忆的信息数量    
  6）   在对话、移动和思考中提高效率    
  7）   允许用户非恶意错误，系统应保护自己不受致命作物的破坏    
  8）   按功能对动作分类，并按此排列屏幕布局，设计者应那里提高命令和动作组织的内聚性    
  9）   提供语境相关的帮助机制

**出错信息和警告是指出现问题时系统给出的坏消息，本系统对于出错信息和警告应该遵循以下原则：**  
  1）   信息以用户可以理解的术语描述；    
  2）   信息应提供如何从错误中恢复的建设性意见；    
  3）   信息应指出错误可能导致那些不量后果，以便用户检查是否出现了这些情况或帮助用户进行改正；    
  4）   信息应伴随着视觉上的提示，如特殊的图像、颜色或信息闪烁。    
  5）   信息不能带有判断色彩，即任何情况下不能指责用户

4.2. 功能性评价(1)易用性:系统的操作简单， 按钮的功能明确，系统界面简话，具有良好的可操作性，用户可以很快上手使用本系统，不需要专业的培训。  
 (2) 可扩展性:系统采用三层架构模式，各层之间职责明确:表示层负贵用户页面的展示:业务逻辑层负责处理整个系统中的业务逻辑规则:数据访问层负责操作数据库:实体层负责将数据封装为对象，作为各层之间数据传递的栽体:基础框架层为系统的运行提供了基础的功能服务。三层架构模式+基础框架，使得本系统具有较强的可扩展性:可以通过扩展基础框架来增如系统运行时底层的基本功能:可以添加实体类，以扩展数据库表结构等数据库对象:可以添加新的业务逻辑类，来处理新的业务逻辑等。  
 (3)安全性:系统采用了多种方式来保证用户在使用系统的过程中，不会产生安全隐患。  
 (4)可维护性:基于三层架构模式+基础框架的代码设计，各层之间相互分离的职责，使得系统的可维护性大大增强:当需要修改获取数据的源头时，只需要修改数据访问层的代码:当需要改变界面呈现的元素样式时，只需要改变表示层的代码;当需要改变整个系统运行的的行为，或者给整个系统添加功能插件、模块时，只需要更改基础框架层的代码等。  
 (5)智能性、健壮性:系统中对于可能出现的人为性错误均进行了自动化的验证处理，比如部门名称不应该重复，用户ID不应该重复等，提高了用户在使用系统时的工作效率，减少了人工对数据进行验证的繁琐步骤，降低了出错的可能性，使得系统更加智能和健壮。

第五章软件设计  
 5.1. 软件设计的六大原则  
 5.1.1.单一职责原则一个软件系统的最佳结构高度依赖于开发这个系统的组织的内部结构。这样，每个软件模块都有且只有一个需要被改变的理由;

一个模块应该只服务于同一类客户，而不是多种客户。例如：有一个工资计算类，在初期由于公司不同部门之间的工资计算方式是一样的。把这个工资计算类用于计算所有部门的工资，将来公司的技术部工资计算方式变了，而其它部门保持不变。就有可能因为修改了这个工资计算类而影响了其它部门的工资计算。应该为不同的客户创建不同的类。

一个函数应该只提供一种功能，不能创建多功能的函数。例如：一个计算函数既可以计算加法又可以计算减法。  
 5.1.2.开放封闭原则  
如果软件系统想要更容易被改变，那么其设计就必须允许增加新的代码来修改系统的行为，而非只能靠修改原来的代码。

也就是说一个类在写完之后就不应该再发生改变（本身业务发生改变和改bug除外）。如果有新的功能加进来，应该可以通过增加新的类来实现。例如：我要实现 “加减乘除” 计算。如果只是简单的写一个类，里面包含“加减乘除”四个函数。这个时候如果需要增加“取余”计算的话，就需要在原来的类上增加“取余”函数。这就违反了开放封闭原则。我们应该创建一个接口，里面包含一个计算函数；然后编写“加减乘除”四个类来实现接口。这样当有“取余”计算方式加进来的时候只需要创建一个新类来实现这个接口就可以了。  
 5.1.3.里氏替换原则  
 如果想用可替换的组件来构建软件系统，那么这些组件就必须遵守同一个约定，以便让这些组件可以相互替换；也就是说子类可以替换父类，且不会对业务产生任何影响。例如父类ParentA 有两个函数Method1 和Method2，子类ChildA继承自父类ParentA且实现了父类的Method1 和Method2函数。而子类ChildB继承自父类ParentA，但是只实现了父类的Method1；这个时候原先的代码是ParentA p = new ChildA();是没有问题的，但是将ChildB替换掉ChildA就可能发生问题了。因为ChildB没有去实现Method2函数；

ChildB中的Method2可能是个空函数；public override void Method2(){} 5.1.4.接口隔离原则  
这项设计原则主要告诫软件设计师应该在设计中避免不必要的依赖。

一个类应该尽量少的把自己的信息透露给客户。应该采用接口的方式来隔离不同的功能函数。例如：ClassA中有20个函数，但是对于同一类客户B只需要用到其中的5个函数，那么应该采用接口Interface的方式来定义这五个函数，让ClassA实现这个接口。这样对于客户B来说就只能看到自己所需要的函数。 5.1.5.依赖倒置原则  
 该设计原则指出高层策略性的代码不应该依赖实现底层细节的代码，两者都应该依赖抽象。

抽象工厂设计模式就是该设计原则最好的实践。使用抽象工厂类来向客户提供相应的功能。当需求发生改变的时候，只需要增加相应的具体实现代码， 并修改抽象工厂类就可以了， 无需更改客户端和原有的具体实现类的代码。  
 5.1.6.迪米特原则

又叫做最少知道原则，如果两个类不必彼此直接通信，那么这两个类就不应当直接调用，可以使用代理的方式通过第三方来转发调用。

一个对象应该对其它对象尽可能的少知道，或者说被引用的对象应该尽量不要把别人不需要的成员暴露出去。这就可以降低类与类之间的耦合度。类之间的耦合度越低就越有利于复用。