



**专业综合拓展**

**基于Spring框架的团队博客系统**

**学 院 计算机学院**

**专 业 软件工程**

**年级班别 2016级(3)班**

**学 号 3116004810**

**学生姓名 陈伟霖**

**指导教师 杨易扬**

**2019年 10 月**

# 摘 要

随着信息化时代的来临，通过团队博客分享技术已经成为一种主流。博客是继Email、BBS和ICQ之后出现的第四种网络交流方式，早期的博客是论坛式的，分享的内容专业度不高，不利于优质内容的寻找、传播及获取，因此由专业团队执笔的私人团队博客成为一种需求。

本论文阐述了开发一个团队博客系统必经的几个阶段，包括该类系统的技术介绍、详细设计、系统实现跟系统测试，解决了在这些阶段开发系统会遇到的问题。本系统后端开发使用JAVA语言、SSM框架，前端开发使用Layui库，通过本论文及上述技术，可以实现该系统。

**关键词：** 团队博客系统，SSM框架，Layui库

# Abstract

In the era of information explosion, sharing the technical information through team blogs has become popular. Blog is the fourth kind of network communication method after Email and ICQ. The early blogs are forum-style, and the sharing content is not professional, which is not conducive to the search, dissemination and acquisition of high-quality content. Therefore, team blogs written by professional teams become a requirement.

This paper describes the stages necessary to develop a team blog system, including Technical introduction, detailed design, system implementation and system test, and solves the problems encountered in the development of the system at these stages. The back-end development of this system uses JAVA language and SSM framework, and the front-end development uses the Layui library. Through this paper and the technology mentioned above, the system can be implemented.

**Key words**：team blog system, SSM framework, Layui library

**目 录**

[1 绪论 1](#_Toc25135161)

[1.1 选题背景 1](#_Toc25135162)

[1.2 选题的目的和意义 1](#_Toc25135163)

[1.3 与当前系统的比较 2](#_Toc25135164)

[1.4 本论文主要工作 2](#_Toc25135165)

[2 技术介绍 3](#_Toc25135166)

[2.1 关系型数据库管理系统介绍 3](#_Toc25135167)

[2.2 SSM框架介绍 3](#_Toc25135168)

[2.3 Layui库介绍 3](#_Toc25135169)

[2.4 开发环境、运行环境、开发工具等介绍 4](#_Toc25135170)

[3 详细设计 5](#_Toc25135171)

[3.1 功能模块图 5](#_Toc25135172)

[3.2 输入输出设计 5](#_Toc25135173)

[3.3 用户界面设计 6](#_Toc25135174)

[3.4 数据库详细设计 8](#_Toc25135175)

[3.4.1 关系模型 8](#_Toc25135176)

[3.4.2 安全性设计 13](#_Toc25135177)

[3.4.3 实体完整性 14](#_Toc25135178)

[3.4.4 参照完整性 14](#_Toc25135179)

[3.4.5 用户自定义完整性 15](#_Toc25135180)

[3.4.5 使用触发器实现更新时间逻辑处理 17](#_Toc25135181)

[4 系统实现 18](#_Toc25135182)

[4.1 数据库实现 18](#_Toc25135183)

[4.1.1 选择存储引擎 18](#_Toc25135184)

[4.1.2 定义索引 18](#_Toc25135185)

[4.1.3 定义数据存放位置 19](#_Toc25135186)

[4.1.4 sql脚本 21](#_Toc25135187)

[4.2 应用程序实现 21](#_Toc25135188)

[4.2.1 权限分离功能实现 21](#_Toc25135189)

[4.2.2 图片上传功能实现 23](#_Toc25135190)

[5 系统测试 27](#_Toc25135191)

[5.1 测试方案 27](#_Toc25135192)

[5.2 测试用例 27](#_Toc25135193)

[6 总结与展望 29](#_Toc25135194)

[参考文献 30](#_Toc25135195)

[附录 31](#_Toc25135196)

# 

# 1 绪论

## 1.1 选题背景

随着信息技术的发展，通过博客分享技术成为当下的一种主流。目前市面上的博客有两种。

第一种是面向群众的大众博客、社区，比如CSDN[1]、博客园[2]，这类博客是属于论坛式的，每个人具有自己的小空间能够发表自己的博客，并且彼此之间能够互相浏览、评价、互动，具有大众性。

第二种是面向私人化的个人/团队博客，比如阿里中间件团队博客[3]、Tgideas腾讯互动娱乐创意设计团队博客[4]等，其文章的编写具有较高专业水平，由团队内部成员编写博客并进行相应的分类，不提供外人编写博客的权利，但保留其评论讨论的功能。



图1.1 阿里中间件团队博客首页

比如在阿里云中间件团队的博客网站上，发表的所有博客均由团队内部人员执笔，该系统提供展示、归档、分类、标签等功能，无游客登录功能。

## 1.2 选题的目的和意义

本次选题是面向专业团队的博客系统，即选题背景中的第二种博客。

本次选题的目的在于为私人团队设计出一种通用的博客系统，使技术人员能够专注于技术分享，因此该论文以实现一个具有权限管理、团队博客编辑、博客展示、游客评论、标签分类的系统为目标，同时后台系统要有相关统计功能。

## 1.3 与当前系统的比较

当前网络上普遍使用的系统，比如前面叙述的Tgideas腾讯互动娱乐创意设计团队博客、阿里中间件团队博客，都存在不同方面的问题。Tgideas腾讯互动娱乐创意设计团队博客界面设计复杂，对浏览者不够友好，评论需要使用QQ进行登录；阿里中间件团队博客采用Hexo模板定制，足够简洁，对浏览者足够友好，但没有评论功能，统计功能也仅有关于博客阅读量、站点访问人数等，缺乏对标签的热度的统计，同时使用hexo编写markdown[5]（以下均简称md）博客，md格式语法需要在线图片的地址（URL），而hexo并不支持图片上传功能。

与当前已有的团队博客比较，我们希望能够设计出一款对浏览者来说足够简洁的、对博客编写者来说功能足够的博客系统。在我们的设计中，博客系统的功能足够简单而强大，访客无需登录，可以自由讨论博文，博客编写者能够根据md语法编写博客，同时系统提供图片上传功能为博客编写提供便利，提供完善的统计功能。

## 1.4 本论文主要工作

本论文分为六大部分，在绪论中介绍选题背景，在技术介绍中简要介绍想要实现该系统应当用到的技术，在详细设计中细化了功能模块图，对输入输出、用户界面进行了设计，数据库方面也进行了详细设计，在系统实现中，讲了数据库的实施与重点难点模块的实现方式，系统测试中完成了对所有功能点的测试，最后一部分对已完成的工作做出了总结跟展望。

# 2 技术介绍

## 2.1 关系型数据库管理系统介绍

本次选用的关系型数据库管理系统（RDBMS）为MySQL[6]。

Mysql是当前最流行的关系型数据库管理系统之一，数据存储遵守关系型数据库的约束，将数据保存在不同的表中，提高了灵活性和增快了速度。MySQL使用标准化的数据库访问语言-- Structure Query Language(SQL)语言，简单易学。同时由于Mysql的开源性，使得其受到大部分中小型企业的喜爱。

## 2.2 SSM框架介绍

SSM框架都是基于JAVA开发的开源框架，分为Spring[7]、SpringMVC[8]、MyBatis[9]三部分。

SpringMVC是基于Model And View(MVC)设计模式的、具有高扩展性的轻量级web框架，Spring通过IOC、AOP两大核心技术解耦，使得开发者能顾专注于应用层的业务逻辑，MyBatis是一款轻量级Object Relational Mapping(orm)框架，其灵活、半自动化的思想既大大简便了开发者的工作，又使得开发者能够为特定的应用场景配置SQL，增加了数据库操作的效率[10]。

## 2.3 Layui库介绍

Layui（谐音：类UI) 是一款采用自身模块规范编写的前端 UI 框架，遵循原生 HTML/CSS/JS 的书写与组织形式，门槛极低，拿来即用[11]。

通过使用Layui库，能够极大简化开发者的工作，节省大量用来调节前端样式的时间，大部分JS代码也不必重新开发，Layui库已经实现并封装好，仅在需要的时候查找对应的API即可。

## 2.4 开发环境、运行环境、开发工具等介绍

开发工具使用eclipse。

项目管理工具软件使用maven，主要用于包管理。

开发语言为JAVA，版本为1.8。

http服务器为TOMCAT，版本为8.0。

MySQL开发环境版本为5.5.53，线上版本为5.7.16，理论上可以兼容5.5-5.7版本中的任意一个版本。

接口开发使用SpringMVC，Spring版本为4.1.7，Mybatis版本使用3.3.0。

开发环境配置为华硕笔记本，win10系统，线上环境为阿里云服务器，linux系统（CentOS 7.2 64位）

# 3 详细设计

## 3.1 功能模块图

在概要设计阶段，我们得到了系统总体的功能模块图，在详细设计阶段，有必要对其进行相应的详细设计和扩展，在整体框架不变的基础上，细分了模块内部的功能，将功能点详细到不可再分，明确每一个模块中有什么样的功能点。

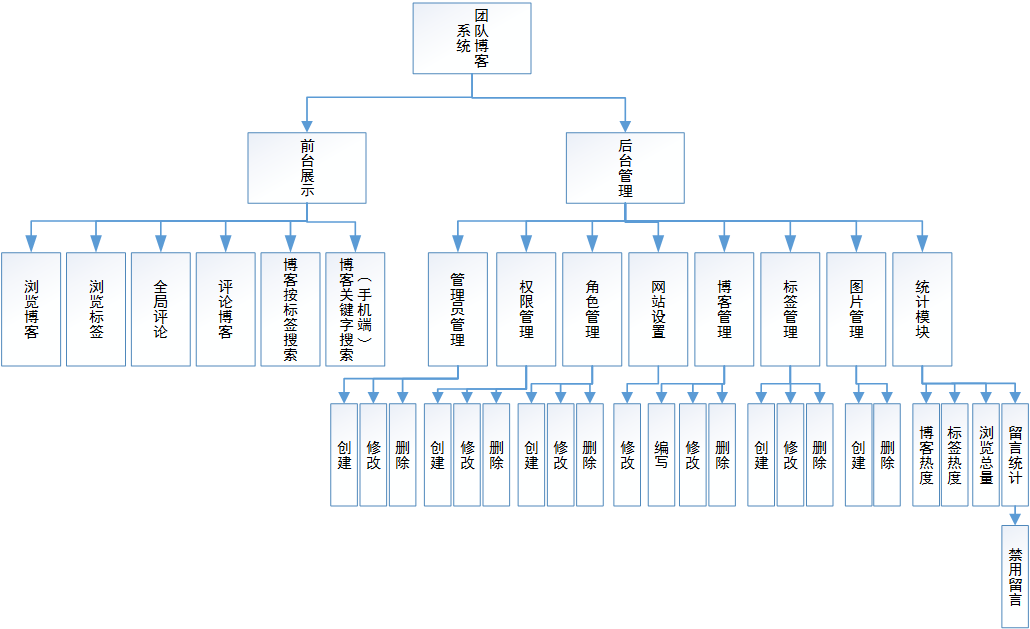


图3.1 详细的功能模块图

## 3.2 输入输出设计

“输入的是垃圾，输出的必然是垃圾”。输入设计的目标是向系统输入正确的数据[12]。

针对编辑博客模块，分析用户操作流程，用户通过点击编辑按钮进入编辑页面，此时页面已有原博客信息，用户只需要修改其中的信息并提交，即使存在未修改的信息，仍然被提交，即提交的数据应当是完整的博客信息，因此输入信息应该包含：

1. 博客标题
2. 博客简介
3. 博客内容
4. 所属标签

以上信息经输入后端处理存入数据库，在处理过程中可能发现不合语义的输入信息，及时提醒用户修改，若输入信息均合法，则跳转到博客列表处，即编辑博客模块的输出信息是：

1. 编辑失败即存在输入数据不合法时，在原页面停留，并提示用户失败信息
2. 编辑成功时，跳转至博客列表，博客列表包含标题、简介、热度、标签、创建时间、最后编辑时间等信息

针对上传图片模块，分析用户操作流程，用户通过点击图片上传按钮进入上传页面，提交所需信息即可，因此输入信息应当包含：

1. 图片描述
2. 图片（大于等于1张图片）

以上信息经输入后，后端对图片进行压缩，为满足不同需求场景，压缩与未压缩的图片均保存在磁盘，写入数据库的为图片描述与压缩、未压缩图片的url，在处理过程中发现图片不合法或者描述信息不合法，应及时提醒用户，通过分析知道，图片上传模块的输出信息是：

1. 上传失败即存在输入数据不合法时，在原页面停留，并提示用户失败信息
2. 上传成功时，跳转至图片列表，图片列表包含图片描述、url、缩略图展示、创建时间等信息

## 3.3 用户界面设计

在输入输出设计的基础上，设计编辑博客模块、上传图片模块的界面，要求界面简洁大方，可见即可得，具体如下：



图3.2 用户界面设计-编辑博客模块

表3.1 数据说明-编辑博客模块

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 数据项名称 | 输入方式 | 长度 | 必输/可选 | 备注 |
| 标题 | 填写 | 50 | 必输 | 博客标题 |
| 简介 | 填写 | 255 | 必输 | 博客简介 |
| 内容 | 填写 | 65535 | 必输 | 博客内容 |
| 所属标签 | 点击选中 | / | 必选 | 博客标签 |

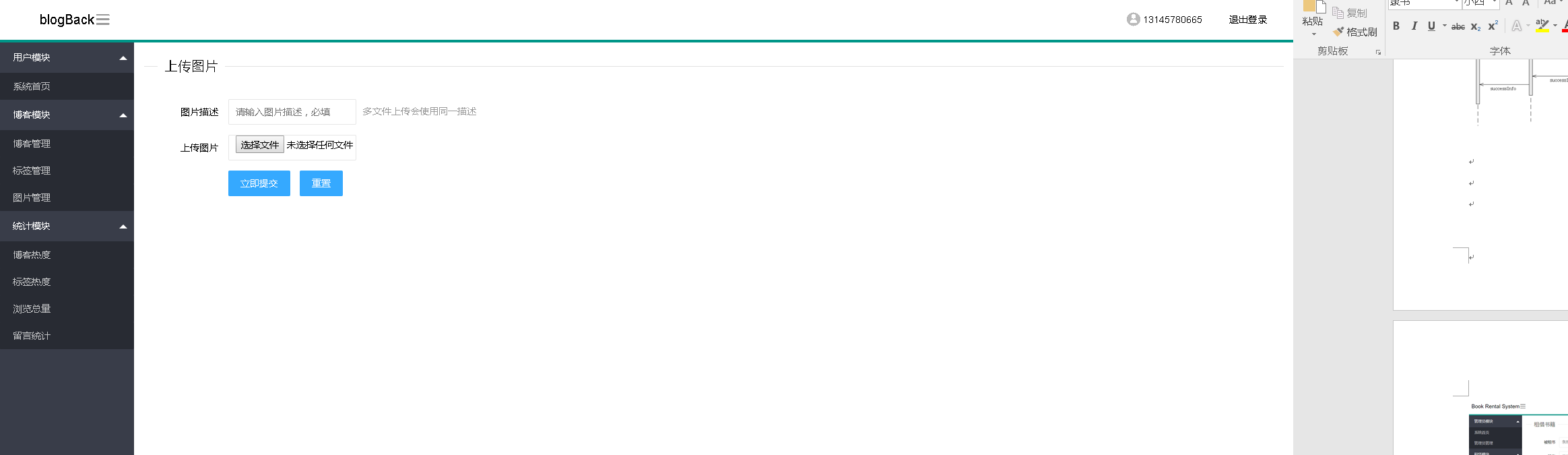


图3.3 用户界面设计-上传图片模块

表3.2 数据说明-上传图片模块

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 数据项名称 | 输入方式 | 长度 | 必输/可选 | 备注 |
| 图片描述 | 填写 | 65535 | 必输 | 无 |
| 上传图片 | 选择框 | / | 必输 | 点击弹出选择框，选择本地图片 |

## 3.4 数据库详细设计

### 3.4.1 关系模型

关系模型从E-R图导出，实体导出成表，属性导出成表的属性，1对1、1对多的联系视情况导出成表或者属性附加在1端、n端，多对多联系导出成表[13]。

同时人为添加某些字段。参考阿里巴巴设计准则[14]，每个表有id、create\_time，update\_time字段，灵活处理，我们仅在大部分表增设这三大字段。理由如下：

增设int类型的自增id字段作为主键，而如果原来已有主键则设置其为unique和not null，这样的好处是提高了查询速度效率。查询的时候对简单、有序的字段进行查询速度最优，倘若以长字符为主键，则会使效率大打折扣。

增设create\_time和update\_time字段是为了更好地维护数据库，增加安全性，防止非法操作，在操作的时候记录时间。

以下是所有关系模式及范式（PK为primary key，FK为foreign key，UN为unique）：

表3.3 admin关系模式

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 名称 | 类型 | 空 | 默认值 | 属性 | 元组语义 | 数据依赖 |
| id | int(11) | 否 | <auto\_increment> | PK |  | admin\_account->id |
| admin\_account | varchar(50) | 否 |  | UN | 管理员帐号 | id->admin\_account |
| admin\_password | char(32) | 否 |  |  | 管理员密码 | id->admin\_password |
| admin\_fullname | varchar(255) | 是 | <空> |  | 管理员姓名 | id->admin\_fullname |
| admin\_phone | char(11) | 是 | <空> |  | 管理员手机号码 | id->admin\_phone |
| status | tinyint(1) | 是 | 2 |  | 状态 1可用 2禁用 默认2 | id->status |
| admin\_ticket | char(32) | 是 | <空> |  | 管理员入场券（用于抢登） | id->admin\_tiket |
| last\_login\_time | int(11) | 是 | <空> |  | 最后登录时间 | id\_last\_login\_time |
| last\_login\_ip | varchar(32) | 是 | <空> |  | 最后登录ip | id->last\_login\_ip |
| create\_time | int(11) | 否 | <空> |  | 创建时间 | id->create\_time |
| update\_time | int(11) | 是 | <空> |  | 更新时间 | id->update\_time |

表3.4 rule关系模式

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 名称 | 类型 | 空 | 默认值 | 属性 | 元祖语义 | 数据依赖 |
| id | int(11) | 否 | <auto\_increment> | PK |  |  |
| rule\_name | varchar(30) | 是 | <空> |  | 权限名称 | id->rule\_name |
| rule | varchar(255) | 是 | <空> |  | 权限规则 | id->rule |
| is\_menu | tinyint(1) | 是 | 2 |  | 是否菜单 1是 2否 默认2 | id->is\_menu |
| parent\_id | int(11) | 是 | <空> | FK | 父级ID 0一级 非0子级 | id->parent\_id |
| icon | varchar(100) | 是 | <空> |  | 图标 | id->icon |
| sort | int(11) | 是 | <空> |  | 排序 | id->sort |
| status | tinyint(1) | 是 | 2 |  | 状态 1可用 2禁用 默认2 | id->status |
| create\_time | int(11) | 否 | <空> |  | 创建时间 | id->create\_time |
| update\_time | int(11) | 是 | <空> |  | 更新时间 | id->update\_time |

表3.5 role关系模式

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 名称 | 类型 | 空 | 默认值 | 属性 | 元祖语义 | 数据依赖 |
| id | int(11) | 否 | <auto\_increment> | PK |  |  |
| role\_name | varchar(30) | 是 | <空> |  | 角色名称 | id->role\_name |
| rule\_ids | varchar(2000) | 是 | <空> |  | 角色权限集合 如1,2,3... | id->rule\_ids |
| status | tinyint(1) | 是 | 2 |  | 角色状态 1正常 2禁用 默认2 | id->status |
| remark | varchar(255) | 是 | <空> |  | 角色注释 | id->remark |
| create\_time | int(11) | 否 | <空> |  | 创建时间 | id->create\_time |
| update\_time | int(11) | 是 | <空> |  | 更新时间 | id->update\_time |

表3.6 admin\_role关系模式

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 名称 | 类型 | 空 | 默认值 | 属性 | 元祖语义 | 数据依赖 |
| id | int(11) | 否 | <auto\_increment> | PK |  | (admin\_id,role\_id)->id |
| admin\_id | int(11) | 否 | <空> | FK | 管理员id | id->admin\_id |
| role\_id | int(11) | 否 | <空> | FK | 角色id | id->role\_id |
| create\_time | int(11) | 否 | <空> |  | 创建时间 | id->create\_time |
| update\_time | int(11) | 是 | <空> |  | 更新时间 | id->update\_time |

除了id作为主键外，联合UN（admin\_id，role\_id）

表3.7 blog关系模式

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 名称 | 类型 | 空 | 默认值 | 属性 | 元祖语义 | 数据依赖 |
| id | int(11) | 否 | <auto\_increment> | PK |  |  |
| title | varchar(255) | 否 |  |  | 博客标题 | id->title |
| description | varchar(255) | 是 | <空> |  | 简介 | id->description |
| content | text | 否 |  |  | 内容 | id->content |
| heat | int(11) unsigned | 否 | 0 |  | 热度 | id->heat |
| create\_time | timestamp | 否 | <INSERT-TimeStamp> |  | 创建时间 | id->create\_time |
| update\_time | timestamp | 是 | <空> |  | 更新时间 | id->update\_time |

表3.8 comment关系模式

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 名称 | 类型 | 空 | 默认值 | 属性 | 元祖语义 | 数据依赖 |
| id | int(11) | 否 | <auto\_increment> | PK |  |  |
| blog\_id | int(11) | 否 | 0 | FK | 所属博客id，0则为全局 | id->blog\_id |
| content | text | 否 |  |  | 评论内容 | id->content |
| name | varchar(50) | 是 | <空> |  | 评论者名字。允许匿名 | id->name |
| mailbox | varchar(50) | 是 | <空> |  | 评论者邮箱，允许匿名 | id->mailbox |
| ip | char(32) | 否 |  |  | 评论者ip | id->ip |
| create\_time | timestamp | 否 | <INSERT-TimeStamp> |  | 创建时间 | id->create\_time |
| status | tinyint(3) unsigned | 是 | 1 |  | 1通过 2下架，默认1，即评论即见 | id->status |

表3.9 tag关系模式

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 名称 | 类型 | 空 | 默认值 | 属性 | 元祖语义 | 数据依赖 |
| id | int(11) | 否 | <auto\_increment> | PK |  |  |
| tag\_name | varchar(30) | 否 |  | UN | 标签名字 | id->tag\_name |
| tag\_content | varchar(255) | 是 | <空> |  | 标签内容 | id->tag\_content |
| create\_time | timestamp | 否 | <INSERT-TimeStamp> |  | 创建时间 | id->create\_time |

表3.10 blog\_tag关系模式：

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 名称 | 类型 | 空 | 默认值 | 属性 | 元祖语义 | 数据依赖 |
| id | int(11) | 否 | <auto\_increment> | PK |  | (blog\_id,tag\_id)->id |
| blog\_id | int(11) | 否 | 0 | FK | 博客id | id->blog\_id |
| tag\_id | int(11) | 否 | 0 | FK | 标签id | id->tag\_id |

除了id主键外，联合UN（blog\_id，tag\_id）

表3.11 img关系模式

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 名称 | 类型 | 空 | 默认值 | 属性 | 元祖语义 | 数据依赖 |
| id | int(11) | 否 | <auto\_increment> | PK |  |  |
| url | varchar(255) | 否 |  |  | 图片url | id->url |
| thumbnail\_url | varchar(255) | 否 |  |  | 压缩图url | id->thumbnail\_url |
| description | text | 否 |  |  | 描述 | id->description |
| create\_time | timestamp | 否 | <INSERT-TimeStamp> |  | 创建时间 | id->create\_time |

表3.12 setting关系模式

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 名称 | 类型 | 空 | 默认值 | 属性 | 元祖语义 | 数据依赖 |
| id | int(11) | 否 | <auto\_increment> | PK |  |  |
| website\_title | varchar(255) | 否 | <空> |  | 网站标题 | id->website\_title |
| copyright | varchar(255) | 是 | <空> |  | 网站版权 | id->copyright |

表3.13 visits关系模式

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 名称 | 类型 | 空 | 默认值 | 属性 | 元组语义 | 数据依赖 |
| id | int(11) | 否 | <auto\_increment> | PK |  |  |
| number | int(11) unsigned | 否 | 0 |  | 网站浏览量 | id->number |

以上关系模式符合属于BCNF。

BCNF定义：设关系模式R<U，F>∈1NF，如果对于R的每个函数依赖X→Y，若Y不属于X，则X必含有超码，那么R∈BCNF。

### 3.4.2 安全性设计

在安全性需求中，我们提到了能够对数据库操作的三类人员：

1. 高级DBA
2. 普通DBA
3. 应用层开发人员

对于这三类用户，我们必须规范他们的权限，给予不同的权限[15]。

对于高级DBA，拥有数据库的全部权限，即我们装数据库时候的root用户，不需要我们再创建。

对于普通DBA，即仅管理当前这个blog数据库，该用户的权限仅有对当前数据库的所有权限，其表、视图、触发器等都由该DBA创建、管理。通过创建用户并授权，语句如下：

1. 创建用户：

CREATE USER 'chenweilinDBA'@'localhost' IDENTIFIED BY 'chenweilin';

2、授予对blog数据库的全部权限：

grant all privileges on blog.\* to chenweilinDBA@'localhost' identified by 'chenweilin'

对于应用层开发人员，我们不允许其随意更改表的结构，其权利仅仅限制在对表的增删改查。通过创建用户比授权：

1. 创建用户：

CREATE USER 'blogDeveloper'@'localhost' IDENTIFIED BY

'weilinCHEN12321+';

2、授予对blog数据库的增删改查权限：

grant select on blog.\* to blogDeveloper@'localhost';

grant insert on blog.\* to blogDeveloper@'localhost';

grant update on blog.\* to blogDeveloper@'localhost';

grant delete on blog.\* to blogDeveloper@'localhost';

grant show view on blog.\* to blogDeveloper@'localhost';

grant execute, trigger on blog.\* to blogDeveloper@'localhost';(执行触发器、存储过程的权利)

### 3.4.3 实体完整性

实体完整性要求每一个表中的主键字段都不能为空或者重复的值。实体完整性指表中行的[完整性](https://baike.baidu.com/item/%E5%AE%8C%E6%95%B4%E6%80%A7/949221)。要求表中的所有行都有唯一的标识符，称为[主关键字](https://baike.baidu.com/item/%E4%B8%BB%E5%85%B3%E9%94%AE%E5%AD%97/1239455)（主码）。

实体完整性的实现在建表的时候就已经完成，因为我们额外使用了id作为表的主码，也就是说所有表的id都是不能为空、不能重复的。在id的设置中，我们使用了primary key代表这是主码，约束了不能为空、不能重复的情况，同时我们利用mysql的特性使id自增，不需要我们关心id为多少，每次插入一条数据会自动生成id且有序，使得重复的情况不会存在，实体完整性得以体现。

当数据违背实体完整性的时候，违约处理应为拒绝操作，比如插入时拒绝数据入库。

### 3.4.4 参照完整性

参照完整性又称引用完整性。参照的完整性要求关系中不允许引用不存在的实体。与[实体完整性](https://baike.baidu.com/item/%E5%AE%9E%E4%BD%93%E5%AE%8C%E6%95%B4%E6%80%A7/1235903)是[关系模型](https://baike.baidu.com/item/%E5%85%B3%E7%B3%BB%E6%A8%A1%E5%9E%8B/3189329)必须满足的完整性约束条件，目的是保证数据的一致性。

参照完整性的实现利用了数据库中的外键，即为foreign key，以下是各个关系模式之间的参照关系：

1、admin\_role关系模式（id, admin\_id, role\_id,。。。）中的admin\_id为外键，参照了admin关系模式的id，role\_id为外键，参照了role关系模式的id。

2、tag\_id关系模式（id, blog\_id, tag\_id）中的blog\_id为外键，参照了blog关系模式中的id，tag\_id为外键，参照了tag关系模式的id。

3、blog关系模式（id, title, admin\_id, 。。。）中的admin\_id为外键，参照了admin关系模式中的id

通过外键的设置，实现mysql中的参照完整性，其违约处理为拒绝操作而非级联操作（级联操作可能会误删某些数据），当应用程序发现数据库拒绝处理的时候应该考虑是否违背了参照完整性。

另外，考虑需求因素，知道有某些参照关系是不需要强一致性的，我们在将其逻辑关系放在程序阶段处理，并将其定为事务防止一致性被破坏。比如：

1、comment关系模式（id, name, blog\_id, 。。。）中的blog\_id如果是0（默认值），则表示为全局留言，不参照其他属性，但输入为非0，则参照了blog关系模式中的id。

2、rule关系模式（id, rule\_name, parent\_id, 。。。）中的parent\_id如果是0（默认），则表示为最高级权限菜单，不参照其他属性，但输入为非0，则参照了自身rule关系模式中的id，表示为其子菜单。

通过应用程序中的事务处理，比如删除id为5的博客时的操作有：删除blog\_tag中的blog\_id=5的元组、删除comment中blog\_id=5的元组、删除blog中id=5的元组，将这三个操作定义为一个事务，能够完成系统对一致性的要求。

### 3.4.5 用户自定义完整性

用户自定义完整性指针对某一具体关系数据库的[约束条件](https://baike.baidu.com/item/%E7%BA%A6%E6%9D%9F%E6%9D%A1%E4%BB%B6/1046571)，它反映某一具体应用所涉及的数据必须满足的语义要求。

1、在各个表中，我们应当对某些字段进行相应的约束，可以使用Mysql的NOT NULL、UNIQUE、默认值进行简单的约束，以下是罗列一些约束：

blog关系模式的title为NOT NULL

blog关系模式的content为NOT NULL

blog关系模式的heat为NOT NULL，默认为0

blog关系模式的admin\_id为NOT NULL

tag关系模式的tag\_name为NOT NULL

具体详细约束，可以参考3.4.1关系模型中导出的关系模式。

2、当数据库中的某些数据约束单单使用NOT NULL、UNIQUE等简单约束无法实现需要的约束的时候，我们考虑使用触发器约束。

①该数据库的大多数关系模式包含了status字段，表示可用/通过、禁用/下架等功能，使用的都是tinyint(1)，尽管已经使用tinyint(1)将其范围约束到0-9，但仍然不够，实际上仅有可用禁用选项时，我们只需要1-2这个范围。因此我们使用触发器[16]来约束。

下面以comment关系模式的status示例：

CREATE TRIGGER comment\_status BEFORE INSERT

ON comment

FOR EACH ROW

BEGIN

DECLARE msg VARCHAR(50);

# 校验status

IF NEW.status <> 1 THEN

IF NEW.status <> 2 THEN

SET msg = 'wrong status number';

SIGNAL SQLSTATE 'HY000' SET MESSAGE\_TEXT = msg;

END IF;

END IF;

END;

由于其他关系模式的status也是同样处理，因此不再展示。

②数据库的admin关系模式中有一个admin\_phone字段，该字段的要求是输入手机号，当然复杂的逻辑我们交给应用程序（比如使用正则校验是否真的为手机号），但是在数据库层面，我们可以限制它的长度为11，倘若插入的不为11我们拒绝插入，因此通过触发器来约束的代码如下：

CREATE TRIGGER admin\_phone BEFORE INSERT

ON admin

FOR EACH ROW

BEGIN

DECLARE msg VARCHAR(50);

# 校验status

IF length(:NEW.admin\_phone) <> 11 THEN

SET msg = 'wrong phone length';

SIGNAL SQLSTATE 'HY000' SET MESSAGE\_TEXT = msg;

END IF;

END;

## 3.4.5 使用触发器实现更新时间逻辑处理

大部分表中都有update\_time字段，即更新时间，即当该表的字段更新时，update\_time自动随之更新，为了实现这个逻辑，我们可以使用触发器。

触发器代码如下：

CREATE TRIGGER blog\_update\_time BEFORE UPDATE

ON blog

FOR EACH ROW

BEGIN

SET NEW.update\_time = now();

END;

# 4 系统实现

## 4.1 数据库实现

### 4.1.1 选择存储引擎

MySQL中的存储引擎有：

1、InnoDB：默认事务存储引擎，5.5版本之后速度更快，实现了事务的ACID特性使得事务奔溃的时候有很好的应对措施，同时支持聚簇索引的建立，提高了性能。

2、MyISAM：提供了大量的特性，包括全文索引、压缩、空间函数(GIS)等，但MyISAM不支持事务和行级锁，有一个毫无疑问的缺陷就是崩溃后无法安全恢复。

3、Archive：存储引擎只支持INSERT和SELECT操作，不是一个事物型的引擎，而是一个针对高速插入和压缩做了优化的简单引擎。

4、Blackhole：不做任何保存，没有存储机制，不考虑。

5、其他引擎：CSV、Federated、Memory、NDB集群等。

对于团队博客系统，显然需要使用到事务的ACID特性，对于事务操作的奔溃处理非常重要，因此我们使用InnoDB存储引擎。

### 4.1.2 定义索引

索引用于快速找出在某个列中有一特定值的行，不使用索引，MySQL必须从第一条记录开始读完整个表，直到找出相关的行，表越大，查询数据所花费的时间就越多，如果表中查询的列有一个索引，MySQL能够快速到达一个位置去搜索数据文件，而不必查看所有数据，那么将会节省很大一部分时间。

索引的定义中我们需要注意，尽量使用小而简单的数据类型比如id，这也是为什么每张表都额外设置一个id的原因，同时为经常需要排序的字段建立索引，为经常进行联合操作的字段建立索引、选择区分度高的字段建立索引。

在MySQL中，由于我们使用的是Innodb存储引擎，其支持索引有：

1、普通索引

2、唯一索引（UNIQUE）

3、主键索引（PRIMARY KEY）

4、组合索引

所有表的主键，MySQL都会自动加上主键索引，即所有id都已经有索引。我们需要考虑的是其他类型的索引。

admin关系模式中的admin\_account建立唯一索引。原因是在查找admin的时候，大部分情况下会根据账户名查找，并且其不允许重复，区分度极高。

admin\_role关系模式中的admin\_id建立普通索引，中间表的作用就是利用admin查询其对应的role，而且区分度也极高。

role关系模式中的role\_name是UNIQUE的，在Mysql中的唯一约束跟唯一索引不加以区分，因此我们需要在这一字段上加上唯一索引。

blog\_tag关系模式中的blog\_id应当建立普通索引，常常使用到的是通过admin\_id查询tag\_id，而admin\_id的区分度很高，重复率很低，tag\_id的区分度比较低，所以我们在blog\_id上建立普通索引。

blog\_tag关系模式中的组合（blog\_id, tag\_id）是不允许重复的，在Mysql中的唯一约束跟唯一索引是不区分的，因此我们必须在这上面建立组合唯一索引。

tag关系模式中的tag\_name是尝试用到的搜索字段，同时还要求唯一，且该字段虽然是varchar但是长度短，因此我们可以加上唯一索引。

### 4.1.3 定义数据存放位置

在开发环境中，数据的存放位置无关紧要，这里我们连接上数据库，通过：show global variables like "%datadir%";

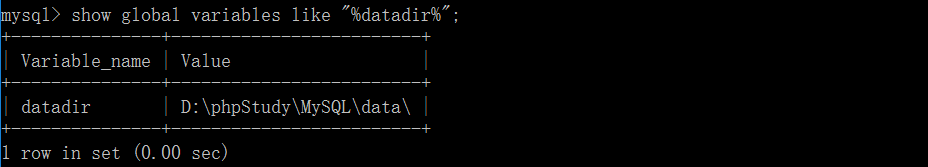


图4.1 显示数据位置

根据指出的路径：

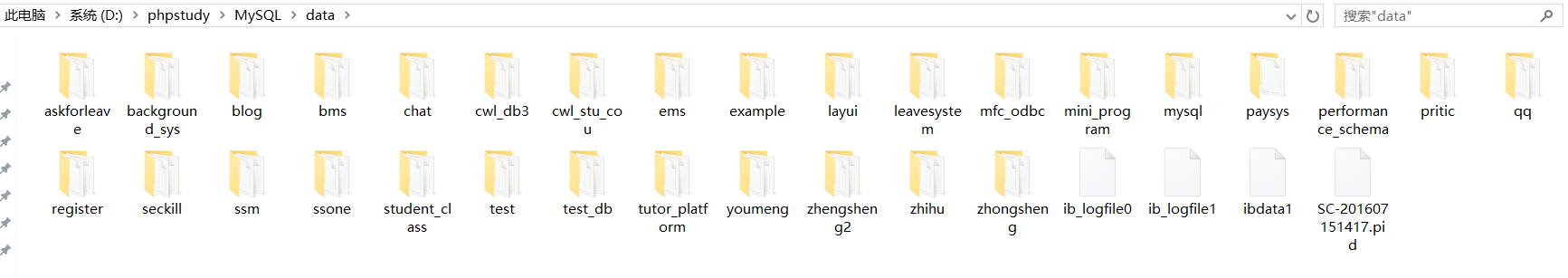


图4.2 数据存放位置

显然，每一个文件夹就是一个表的物理存储位置。

在生产环境中，数据存储盘会独立开来，默认安装的MySQL的数据存放位置需要重新规划修改，储存到不同位置。

下面是线上数据库的位置：

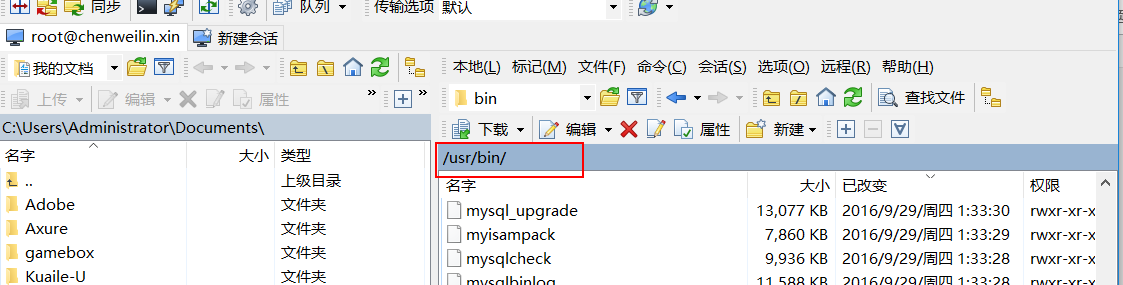


图4.3 生产环境的数据库位置

下面是线上数据库数据存放位置：

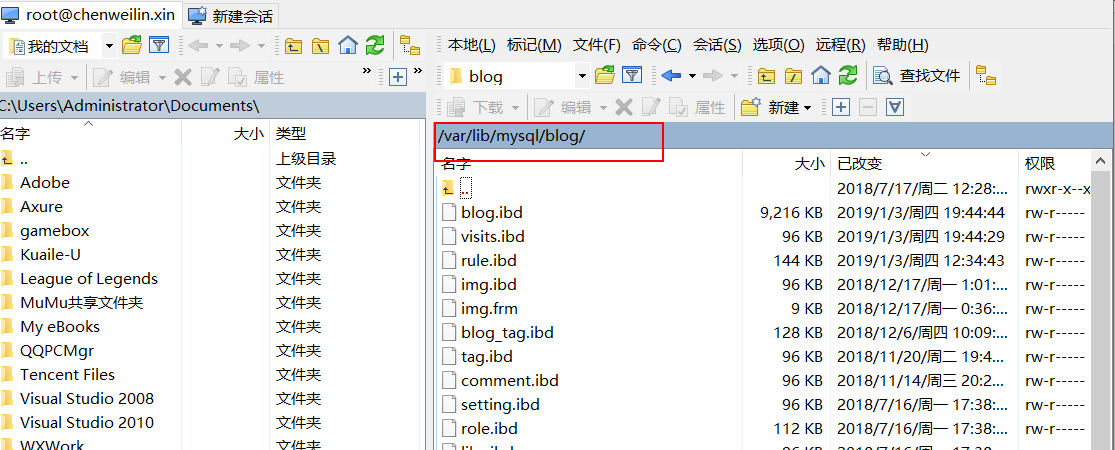


图4.4 生产环境的数据存放位置

明显看到，生产环境已经实现了数据库与数据存储位置的分离。

### 4.1.4 sql脚本

见附录。

## 4.2 应用程序实现

### 4.2.1 权限分离功能实现

数据库层面的安全性通过数据库用户user的授权来维护，而应用层面上的安全性则要通过权限分离来维护。

为应用面向的每个用户进行权限的规定和授予，这里需要用到数据库中的admin、role、rule、admin\_role表进行。具体流程为：

当管理员登陆的时候，后端根据账号密码确认通过，通过时查询数据库中存储的该管理员的权限集合写入session中，当管理员进行某一次操作的时候都校验session中的权限集合，如果不存在该权限则表明无此权限，应当拒绝操作。流程图[17]表述如下：

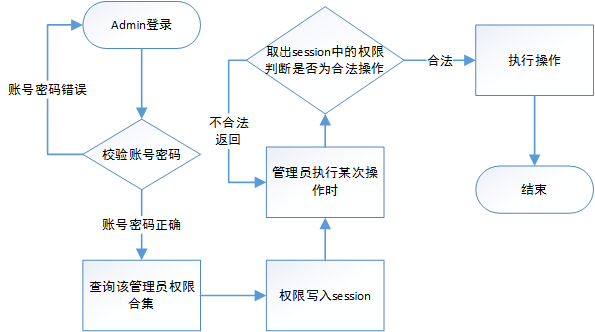


图4.5 admin执行操作时系统逻辑处理

核心代码如下：

**private** **static** **final** String[] ***IGNORE\_URI*** = {"login", "logout", "/static/", "admin/404", "/frontApi/"};

@Override

**public** **boolean** preHandle(HttpServletRequest request, HttpServletResponse response, Object handler)

**throws** Exception {

//flag means if it's needed for intercepted, default needed

**boolean** flag = **false**;

String servletPath = request.getServletPath();

**for** (String s : ***IGNORE\_URI***) {

**if** (servletPath.contains(s)) {

//不需要拦截的uri

**return** **true**;

}

}

//如果拦截，首先判断是否有权限

**if** (!flag) {

Admin admin = (Admin) request.getSession().getAttribute("admin");

List<Rule> ruleList = admin.getRuleList();

**for** (Rule rule : ruleList) {

**if** (servletPath.contains(rule.getRule())) {

flag = **true**;

}

}

}

//没有权限的操作

**if** (!flag) {

response.setContentType("text/html;charset=utf-8");

PrintWriter out = response.getWriter();

out.print("<script language='javascript'>alert('您无此操作权限，请联系管理员');window.location.href='" +

request.getHeader("Referer") + "';</script>");

//重定向到上一个url

}

//有权限则返回true，对操作放行

**return** flag;

}

### 4.2.2 图片上传功能实现

图片上传时，系统校验图片格式，然后处理图片，防止存储时相同名字的图片覆盖，然后生成缩略图，图片与缩略图一并保存在磁盘中供不同需求使用，图片url与描述写入数据库，因为图片比较大，如果不保存在磁盘而直接写入数据库，会给数据库带来巨大的负担。流程图[5]表述如下：

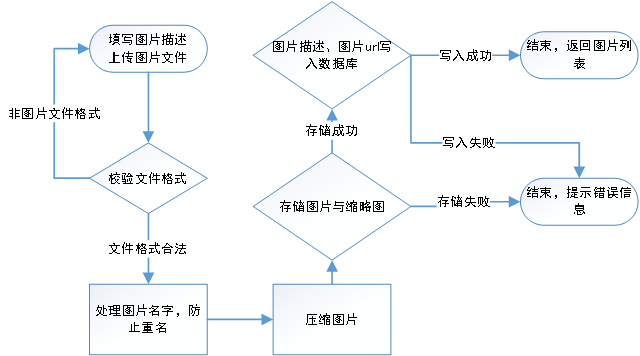


图4.6 上传图片逻辑处理

核心简要代码如下：

@Override

**public** **void** uploadImg(String parentPath, ImgForm imgForm) **throws** InputWrongFormatException, BaseException {

List<MultipartFile> files = imgForm.getFiles();

**if** (files == **null** || files.size() <= 0) {

**throw** **new** InputWrongFormatException("必填字段不能为空");

}

List<String> fileNames = **new** ArrayList<String>();

**for** (MultipartFile multipartFile : files) {

//判断是否为图片格式，如果不是则报错

InputStream inputStream = **null**;

**try** {

inputStream = multipartFile.getInputStream();

**if** (!ValidatorUtil.*isImage*(inputStream)) {

**throw** **new** InputWrongFormatException("文件格式错误");

}

} **catch** (IOException e1) {

**throw** **new** BaseException("系统繁忙，稍后再试");

} **catch** (InputWrongFormatException e) {

**throw** e;

} **finally** {

**…** //关闭流

}

}

//客户端上传的文件的名字，需要防止重名，加上时间戳和随机生成的随机长度字符串

String avoidRepeat = **new** Date().getTime() + MyStringUtils.*getRandomString*(**new** Random().nextInt(10));

String fileName = avoidRepeat + multipartFile.getOriginalFilename();

fileNames.add(fileName);

File imgFile = **new** File(parentPath + "../blogImg", fileName);

File thumbnailsFile = **new** File(parentPath + "../blogImg/thumbnail", fileName);

**try** {

//保存图片和压缩图片

multipartFile.transferTo(imgFile);

ImgUtils.*compressImg*(imgFile, thumbnailsFile);

} **catch** (Exception e) {

**throw** **new** BaseException("文件上传错误，请稍候再试");

}

}

//获取url包装成entity.Img的List

String description = imgForm.getDescription();

List<Img> imgList = **new** ArrayList<Img>();

**for** (String name : fileNames) {

imgList.add(**new** Img(SystemConstantEnum.***PRE\_IMG\_URL*** + "/" + name, SystemConstantEnum.***PRE\_IMG\_URL*** + "/thumbnail/" + name, description));

}

//数据库操作

**try** {

imgDao.addImgs(imgList);

} **catch** (Exception e) {

//数据库操作错误的情况下，为了节省服务器静态资源负担，应该把

//没有写入数据库而存在服务器上的文件删除

**for** (Img i : imgList) {

**new** File(i.getUrl()).delete();

**new** File(i.getThumbnailUrl()).delete();

}

**throw** **new** BaseException("系统繁忙，稍后再试");

}

}

# 5 系统测试

## 5.1 测试方案

测试方法：黑盒测试，通过测试用例（包括正确的输入和错误的输入及操作）对系统的所有功能模块进行操作，由系统的执行结果是否一致来判断系统的功能是否实现并且是完善的。

选用测试用例的原则：测试用例要包含到系统的所有功能；测试用例要包含正确的和错误的两种输入/操作。

## 5.2 测试用例

部分测试用例如下，T/F表示期望输出与实际情况是否一致：

表5.1 部分测试用例

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 窗体名称 | | 功能 | 输入/动作 | 期望的输出 | 实际情况 | T/F |
| 手机端  展示页面 | 首页 | 展示博客列表 | 进入首页 | 博客列表 | 博客列表 | T |
| 搜索层 | 查询博客 | 输入JAVA | JAVA相关博客列表 | JAVA相关博客列表 | T |
| 空 | 所有博客列表 | 所有博客列表 | T |
| 全局留言 | 全局留言 | 输入留言内容 | 留言成功 | 留言成功 | T |
| 输入留言内容和邮箱 | 留言成功，发送邮件 | 留言成功，发送邮件 | T |
| 空 | 留言失败，提示需要输入内容 | 留言失败，提示需要输入内容 | T |
| 博客留言 | 博客下留言 | 输入留言内容 | 留言成功 | 留言成功 | T |
| 输入留言内容和邮箱 | 留言成功，发送邮件 | 留言成功，发送邮件 | T |
| 空 | 留言失败，提示需要输入内容 | 留言失败，提示需要输入内容 | T |
| 电脑端展示页面 | 首页 | 展示博客列表、部分留言、所有标签 | 进入首页 | 博客列表、部分留言、所有标签 | 博客列表、部分留言、所有标签 | T |
| 标签搜索 | 点击首页标签 JAVA | JAVA标签的博客列表 | JAVA标签的博客列表 | T |
| 全局留言 | 全局留言 | 输入留言内容 | 留言成功 | 留言成功 | T |
| 输入留言内容和邮箱 | 留言成功，发送邮件 | 留言成功，发送邮件 | T |
| 空 | 留言失败，提示需要输入内容 | 留言失败，提示需要输入内容 | T |
| 博客留言 | 博客下留言 | 输入留言内容 | 留言成功 | 留言成功 | T |
| 输入留言内容和邮箱 | 留言成功，发送邮件 | 留言成功，发送邮件 | T |
| 空 | 留言失败，提示需要输入内容 | 留言失败，提示需要输入内容 | T |

详尽版测试用例，见附录。

# 6 总结与展望

基于前篇论文截止到概要设计分析，本论文继续分析并完成了团队博客系统的后续开发。在本论文绪论中介绍选题背景，在技术介绍中简要介绍想要实现该系统应当用到的技术，在详细设计中细化了功能模块图，对输入输出、用户界面进行了设计，数据库方面也进行了详细设计，在系统实现中，讲了数据库的实施与重点难点模块的实现方式，系统测试中完成了对所有功能点的测试，最后一部分对已完成的工作做出了总结跟展望。可以认为本论文已经完成了开发一个系统的后期工作。

本次开发的系统具有权限分离作用，能够使团队内部人员各司其职，仅完成自己应该完成的任务并且不越权操作，在页面展示方面比较美观，符合页面开发的基本要求。但仍有不足，比如最开始的时候构想有一个抢登功能还未实现，前台展示页面可以考虑开发出一个讨论社区，提供注册登录功能而不仅仅是游客浏览和游客留言，使得功能更加齐全和博客站更加吸引人。当前开发完成的博客上线前台展示页面为https://chenweilin.xin，在未来会不断加入上述功能，使该博客系统更加完善。

# 参考文献

[1] 世纪乐知. CSDN-专业IT社区[DB/OL] . https://www.csdn.net/,2019-10-27

[2] 博客园. 博客园-开发者的网上家园[DB/OL] . ttps://www.cnblogs.com/,2019-10-27

[3] 阿里中间件团队. 阿里中间件团队博客[DB/OL] . http://jm.taobao.org/,2019-10-27

[4] Tgideas. Tgideas-腾讯互动娱乐创意设计团队[DB/OL] . https://tgideas.qq.com/,2019-10-28

[5] HarryZhu. 文档定义应用：数据科学的文档革命[DB/OL] . https://segmentfault.com/a/1190000005125984,2019-10-28

[6] ORACKE. MySQL[DB/OL] . https://www.mysql.com/,2019-10-28

[7] Pivotal. Spring[DB/OL] . https://spring.io/,2019-10-28

[8] Pivotal. Web on Servlet Stack[DB/OL] . https://docs.spring.io/spring/docs/current/spring-framework-reference/web.html,2019-10-28

[9] MyBatis. mybatis-Mybatis 3|Introduction[DB/OL] . https://mybatis.org/mybatis-3/index.html,2019-10-28

[10] 王天顺,程杰,张志文. MyBatis在《Web中间件技术》课程中的应用[J] . 教育教学论坛,2019(38):204-205.

[11] 贤心. 经典模块化框架[DB/OL] . https://www.layui.com/doc/,2019-10-28

[12] 王晓敏.信息系统分析与设计[M].4 版.北京：清华大学出版社，2013：248-252.

[13] 姗, 王, and 萨 诗煊. 数据库系统概论[M] 北京, 高等教育出版社, 2014：205-420.

[14] 杨冠宝. 阿里巴巴Java开发手册v1.4.0（详尽版）[M]. 北京, 电子工业出版社, 2018：25-29.

[15] J. Murach. murach’s MySQL[M] . WOW! eBook & Associates，2015(2nd edition)： 492-522

[16] B. Forta. Sams Teach Yourself SQL in 10Minutes[M]. United States of America, Sams, 2012: 204-205

[17] 张海藩. 软件工程导论[M]. 北京：清华大学出版社， 2003：124-125

# 附录

附文件：blog.sql，是sql代码文件

附文件：测试报告.docx，包含了完整的测试报告

附文件：系统安装说明书.docx，详细阐述了如何在本地使用该系统

附源程序：blogBack、dist