**数据库课程设计报告**

--------个人博客管理系统

学院：软件学院

专业：网络空间安全

目录

1. 系统开发平台····································3
2. 数据库规划·········································3

2.1任务陈述········································3

2.2任务目标········································3

1. 系统定义···········································3

3.1系统边界·······································3

3.2用户视图·······································4

1. 需求分许··········································4

4.1用户需求说明···································4

4.1.1数据需求··································4

4.1.2事务需求··································4

4.2系统需求说明···································5

1. 数据库逻辑设计·····································5

5.1 E-R图···········································5

5.2数据字典········································6

1. 数据库物理设计·····································7

6.1安全机制········································7

6.2其他············································7

1. 应用程序设计·······································8

7.1功能模块设计····································8

7.2界面设计······································20

7.3事务设计······································24

1. 测试与运行·······································29
2. 总结··············································31

附：参考文献··········································33

1. **系统开发平台：**

题目：个人博客管理系统

开发工具：IDEA

开发环境：Maven+jdk1.8

数据库：MySQL

操作系统：window10

1. **数据库规划**

**2.1任务陈述：**

个人博客管理系统旨在实现用户对自己各种博客信息的统一管理，使用户能够随时发布、查询、更改个人的博客信息，实现对自己博客的管理。系统的开发实现了系统化、规范化和自动化。

**2.2任务目标：**

实现个人博客的随时管理，系统操作应该简单易用，适合各个年龄段的用户。

实现用户信息的验证

实现用户个性化的页面设计

实现博客类型唯一性的检验

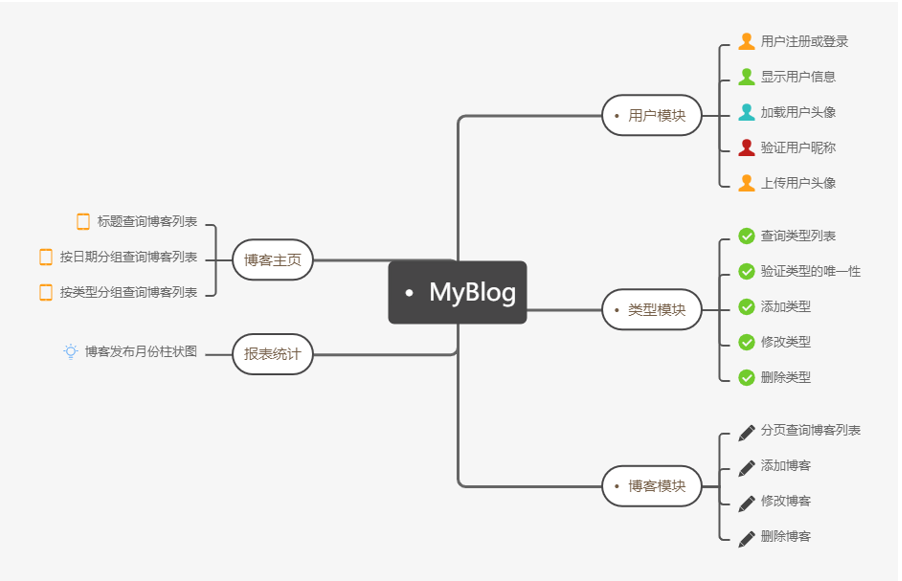
实现用户对博客信息的增删改查操作

实现用户博客的分页展示

实现每个用户每月的博客报表统计

1. **系统定义**

**3.1系统边界：**



3.2用户视图：

首先每个用户会到的登录页面进行登录，当后台验证成功后，页面就会跳转到系统首页，在这个页面，用户会看到自己已经发布过的博客目录，点击链接就会出现博客详情。最上面一行就是系统导航栏，用户可以根据自己的需求，进入相应的页面进行操作。而每个变化的操作界面都是在一个容器中，根据不同的请求操作跳转到不同的操作页面。左边一列是多种博客查询手段，用户可根据自己的需要实现快速查询。

1. **需求分析**

**4.1用户需求分析**

4.1.1数据需求：

在博客管理系统中，对于每个不同的用户来说，他们都有唯一标识ID，通过该必要属性，他们就能够查询到自己的博客信息，并且对他们做出相应的增删改查操作。而每篇博客作为一个独立的实体，他们也有互相之间唯一识别的属性blogId。为了满足数据库BC范式的需求，减少在大量数据时产生的冗余，将博客的类型属性typeId从blog实体中脱离出来，成为第三大实体。这三大实体之间的数据流动，形成了该博客管理系统。

4.1.2事务需求：

1. 用户的注册和登录
2. 查询用户信息
3. 加载/上传用户头像
4. 验证用户昵称
5. 验证博客类型的唯一性
6. 博客类型的增删改查
7. 博客内容的增删改查
8. 按照不同的查询方式实现博客的精准查询

**4.2系统需求说明：**

**4.2.1初始数据库大小：**

博客总共有220条。

博客类型总共有20中。

顾客数量总共有36人。

**4.2.2数据库增长速度：**

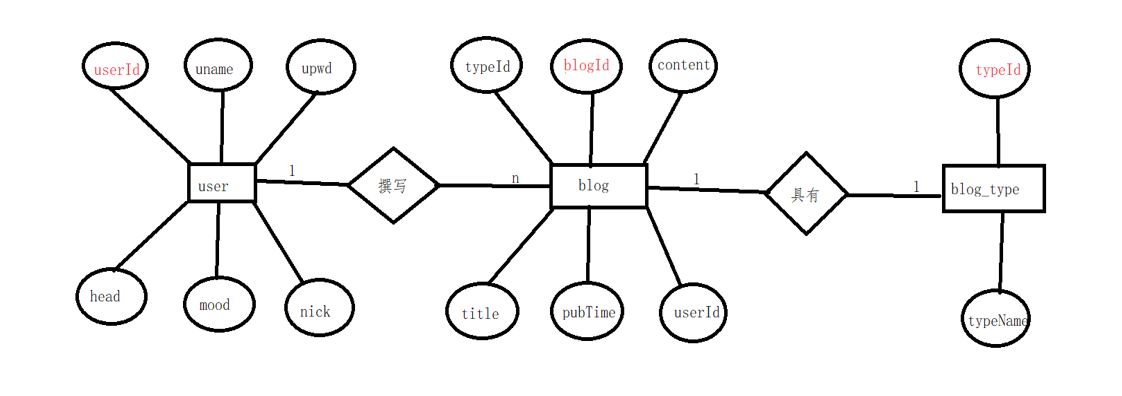
每天大约有5-6人注册博客，需要添加用户信息。

每个用户每天发布1-2篇博客，需要同时添加博客信息和博客类型信息。

**4.2.3安全性：**

1. 数据库登陆密码必须进行加密存储。
2. 通过filter过滤器，对所有用户的数据请求行为进行过滤，使得每个用户只能他么自己的博客空间内进行相应的数据操作。
3. **数据库逻辑设计**

**5.1 E-R图：**



**5.2数据字典：**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 实体 | 属性 | 描述 | 键 | 类型 | 可否为空 | 是否多值 |
| 博客  (blog) | blogId | 博客Id | 主键 | int | 否 | 否 |
| title | 博客标题 | / | varchar | 否 | 否 |
| content | 博客内容 | / | text | 是 | 否 |
| typeId | 博客类型 | 外键 | int | 否 | 否 |
| pubTime | 发布时间 | / | timestamp | 否 | 否 |
| userId | 用户Id | 外键 | int | 否 | 否 |
| 博客类型  （blog\_type) | typeId | 博客类型 | 主键 | int | 否 | 否 |
| typeName | 类型名称 | / | varchar | 否 | 否 |
| 用户  （user) | userId | 用户Id | 主键 | int | 否 | 否 |
| uname | 用户姓名 | / | varchar | 否 | 否 |
| upwd | 用户密码 | / | varchar | 否 | 否 |
| nick | 用户昵称 | / | varchar | 否 | 否 |
| Head | 用户头像 | / | varchar | 否 | 否 |
| Mood | 个性签名 | / | varchar | 否 | 否 |

创建blog表：

Create table blog(

blogId int not null,

Title varchar(50) not null,

Content text,

typeId int primary key,

pubTime timestamp not null,

userId int primary key)

创建blog\_type表：

Craete table blog\_type(

typeId int not null,

typeName varchar(50) not null)

创建user表：

Create table user(

userId int not null,

Uname varchar(50) not null,

Upwd varchar(50) not null,

Nick varchar(50),

Head varchar(100),

Mood varchar(500));

1. **数据库物理设计**

**6.1索引**

1. blog表上以title创建索引：

Create index IX\_title on blog(title);

1. blog\_type表上以blogName创建索引

Create index IX\_blogName on blog\_type(blogName);

1. user表上以uname创建索引

Create index IX\_uname on user(uname);

**6.2安全机制**

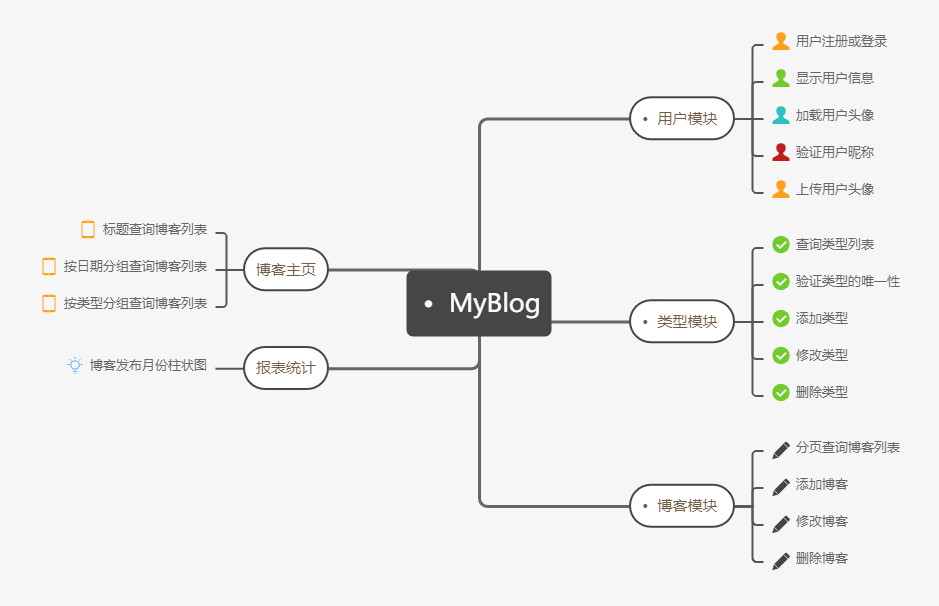
**数据安全：**

1. 系统登录时需要经过SQL Sever登录认证，登陆的账号和密码正确才能启动程序。
2. 用户密码是经过MD5加密，以密文的形式存储在数据库中。
3. 所有的SQL语句都是以预备语句的形式执行，极大程度上降低了SQL注入的风险。

**系统安全：**

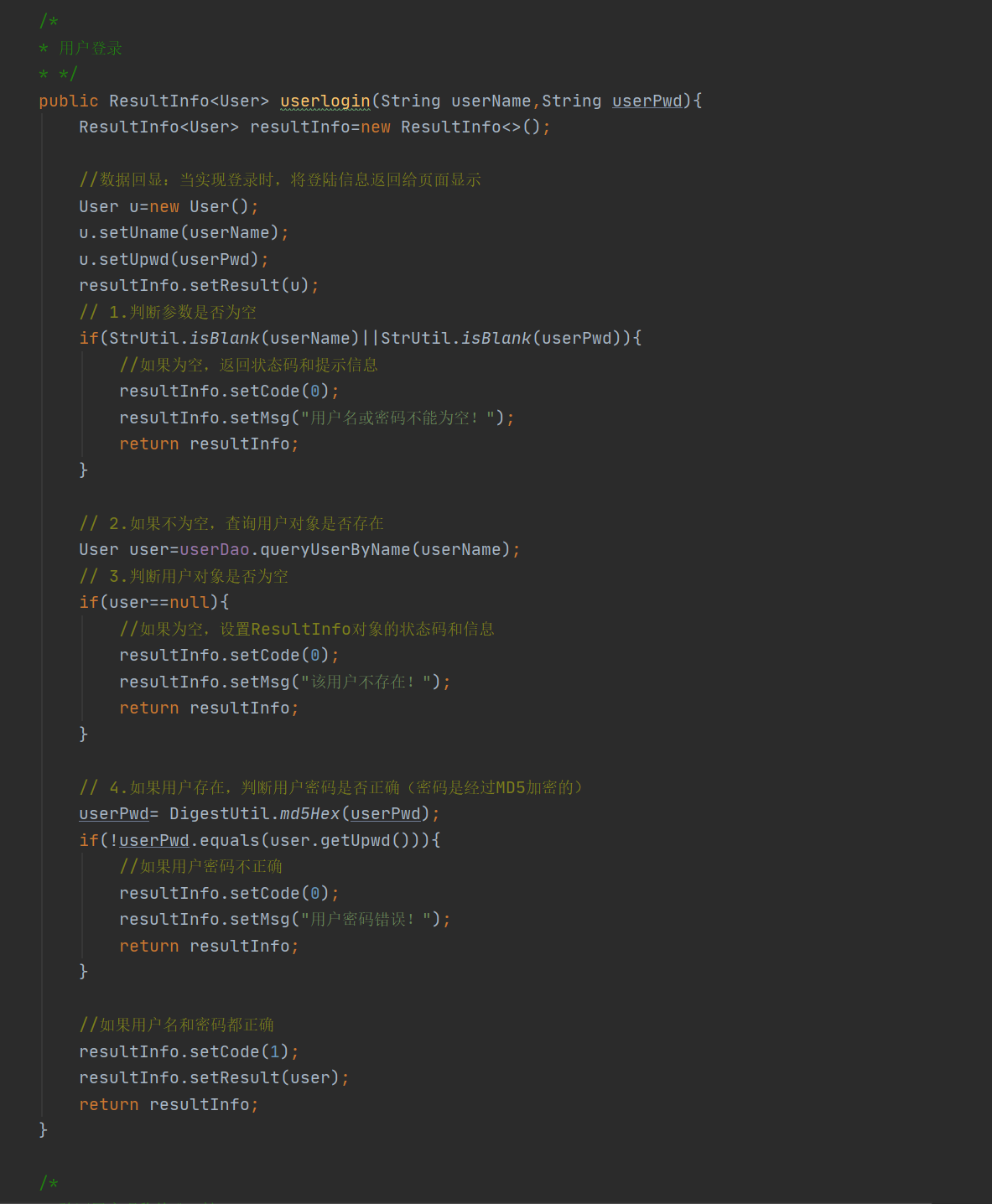
1. 登录系统时必须经过后台身份认证，当户名和密码完全正确时才能够登陆成功。
2. 系统后台通过filter过滤器，过滤所有的用户行为，使不同的用户只能查看自己的博客页面。
3. **应用程序设计**

**7.1功能模块：（只展示部分重要代码）**

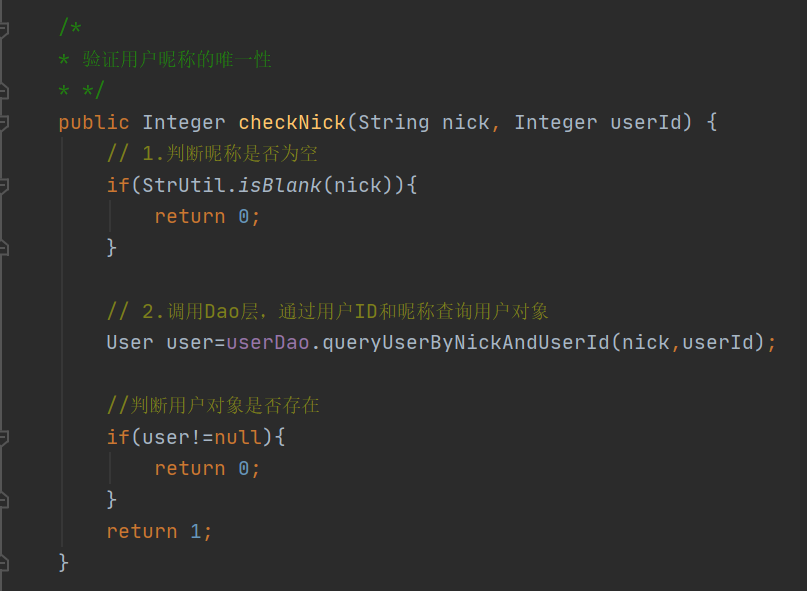


用户模块：

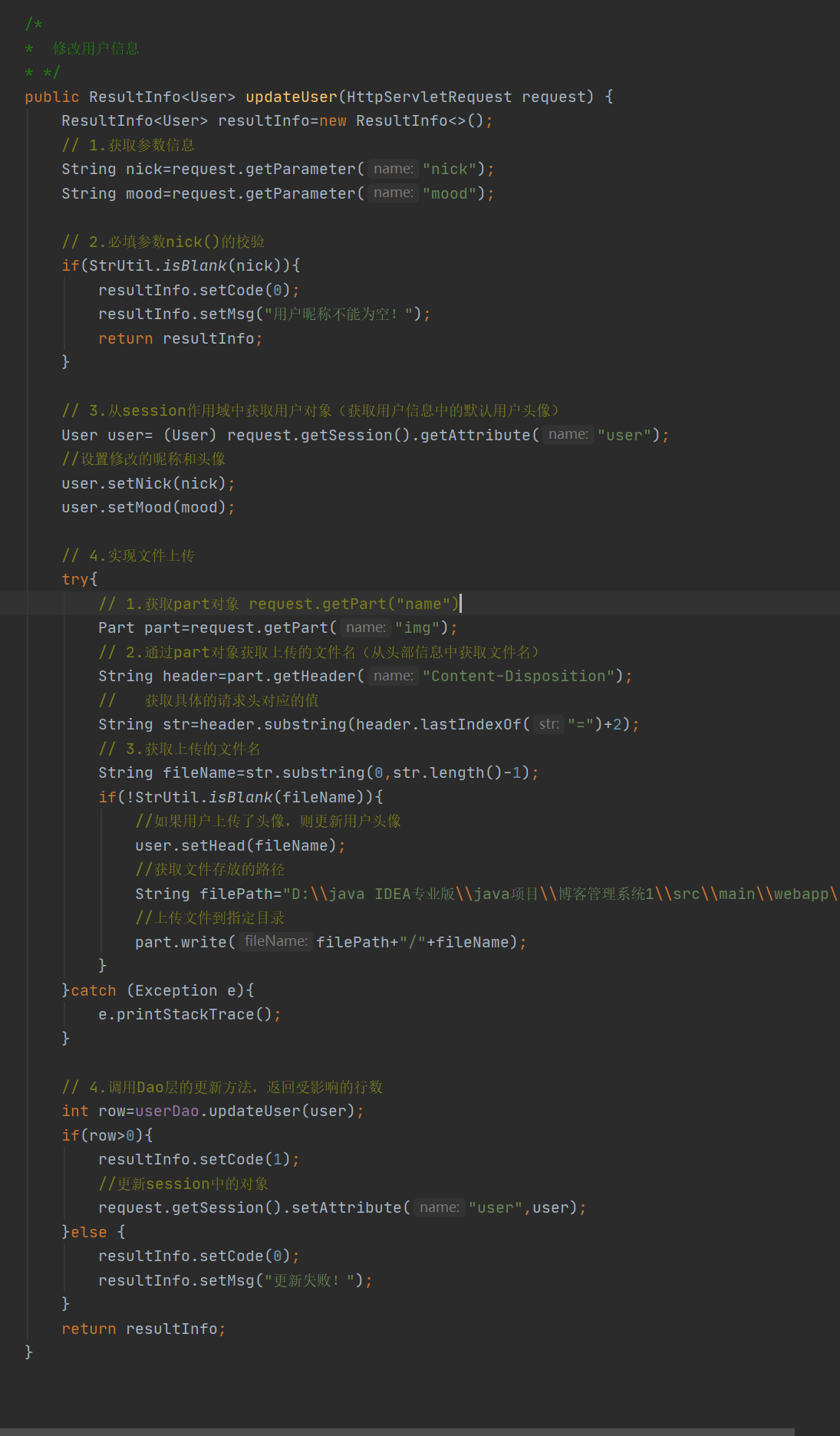
1. 用户登录：



1. 用户昵称唯一性的验证：

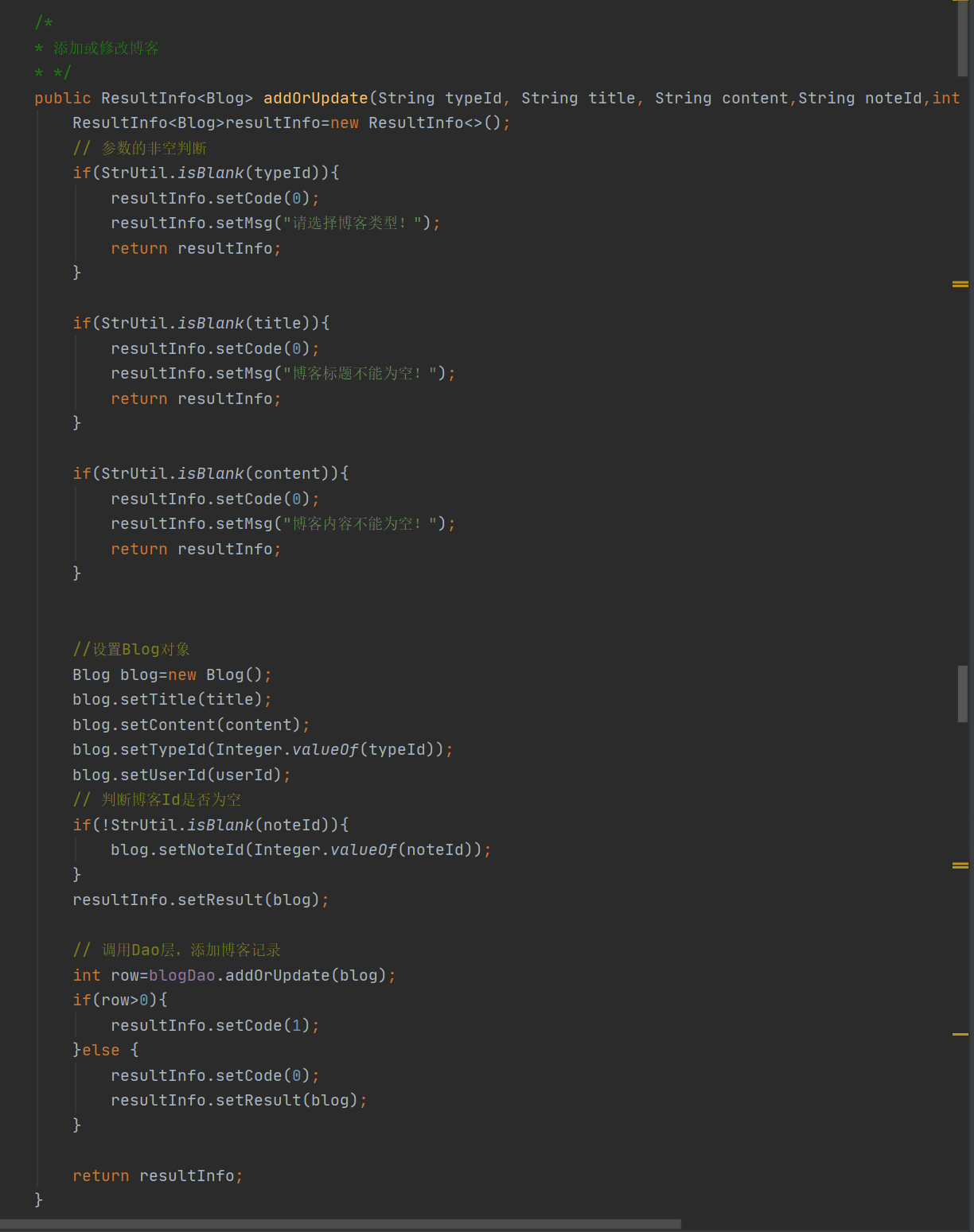


1. 修改用户个人信息：

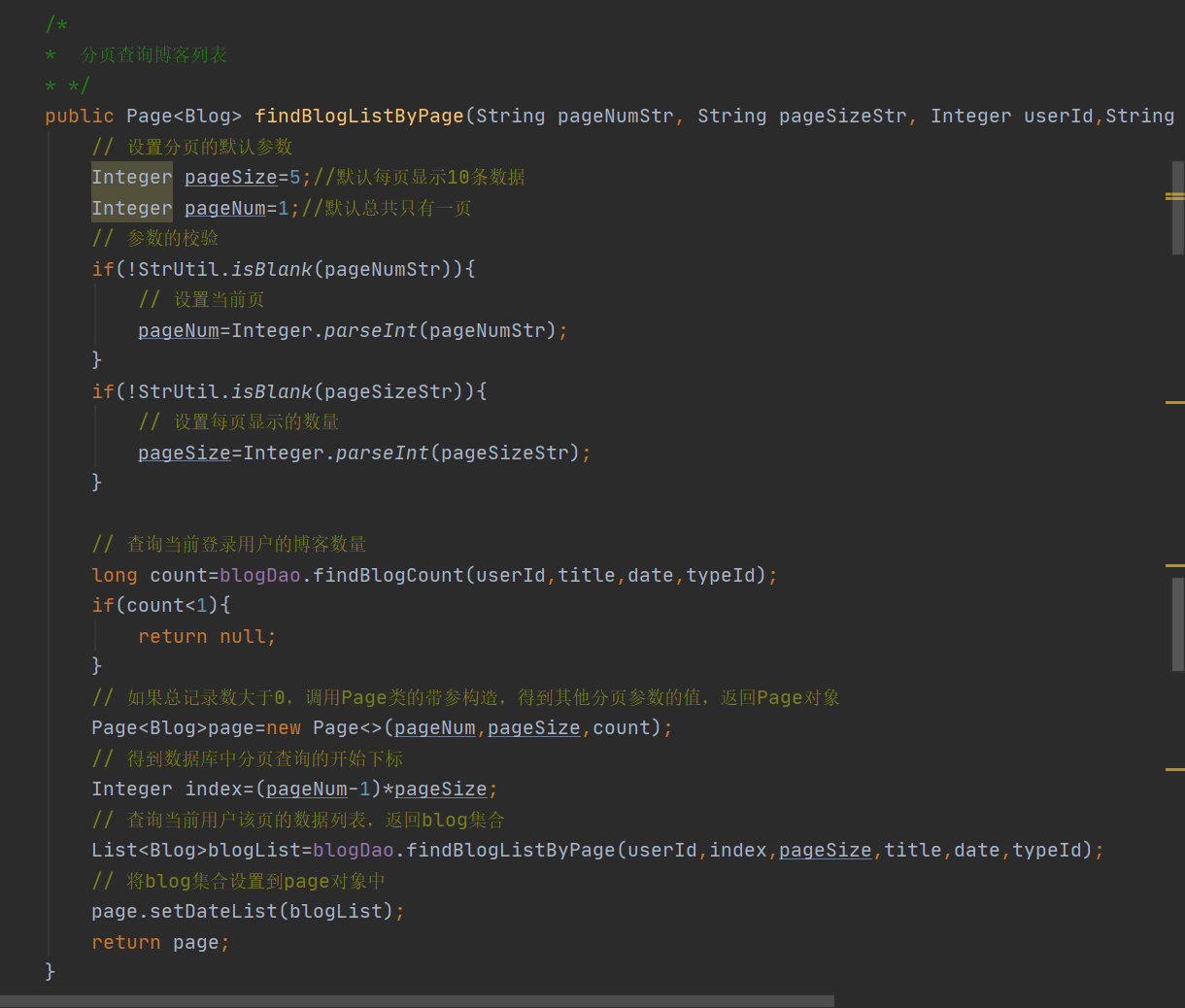


博客模块：

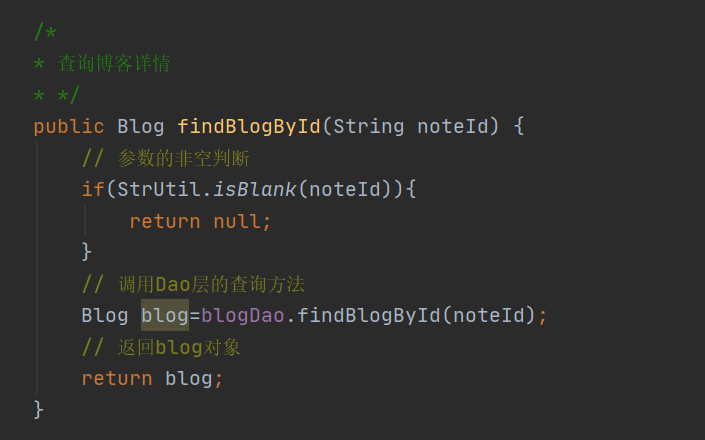
1. 添加或修改博客：



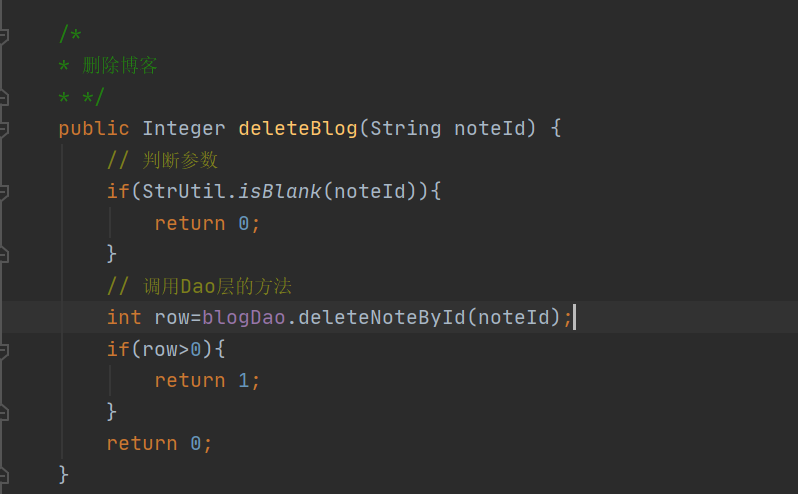
1. 分页查询博客列表：



1. 博客详情展示：



1. 删除博客：

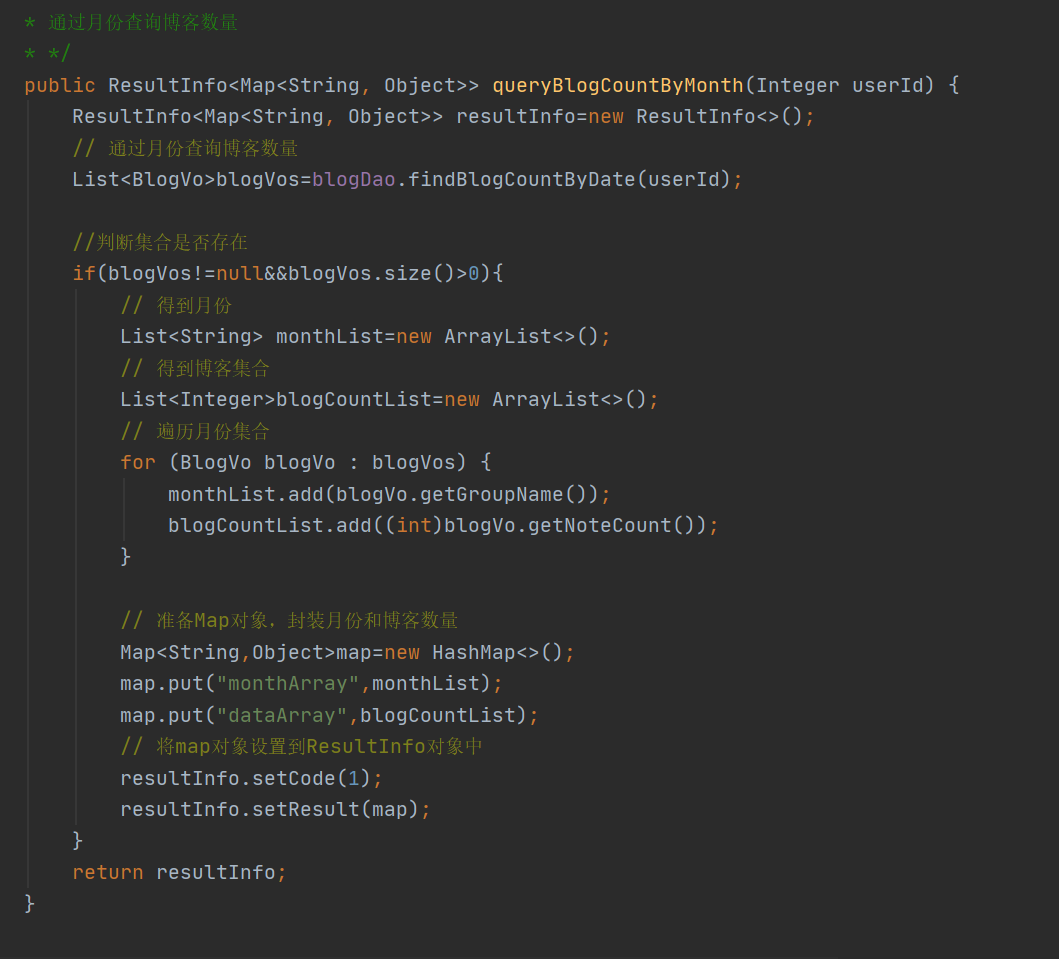


博客主页模块：

1. 不同方式查询博客数量：



1. 按照月份查询博客数量：

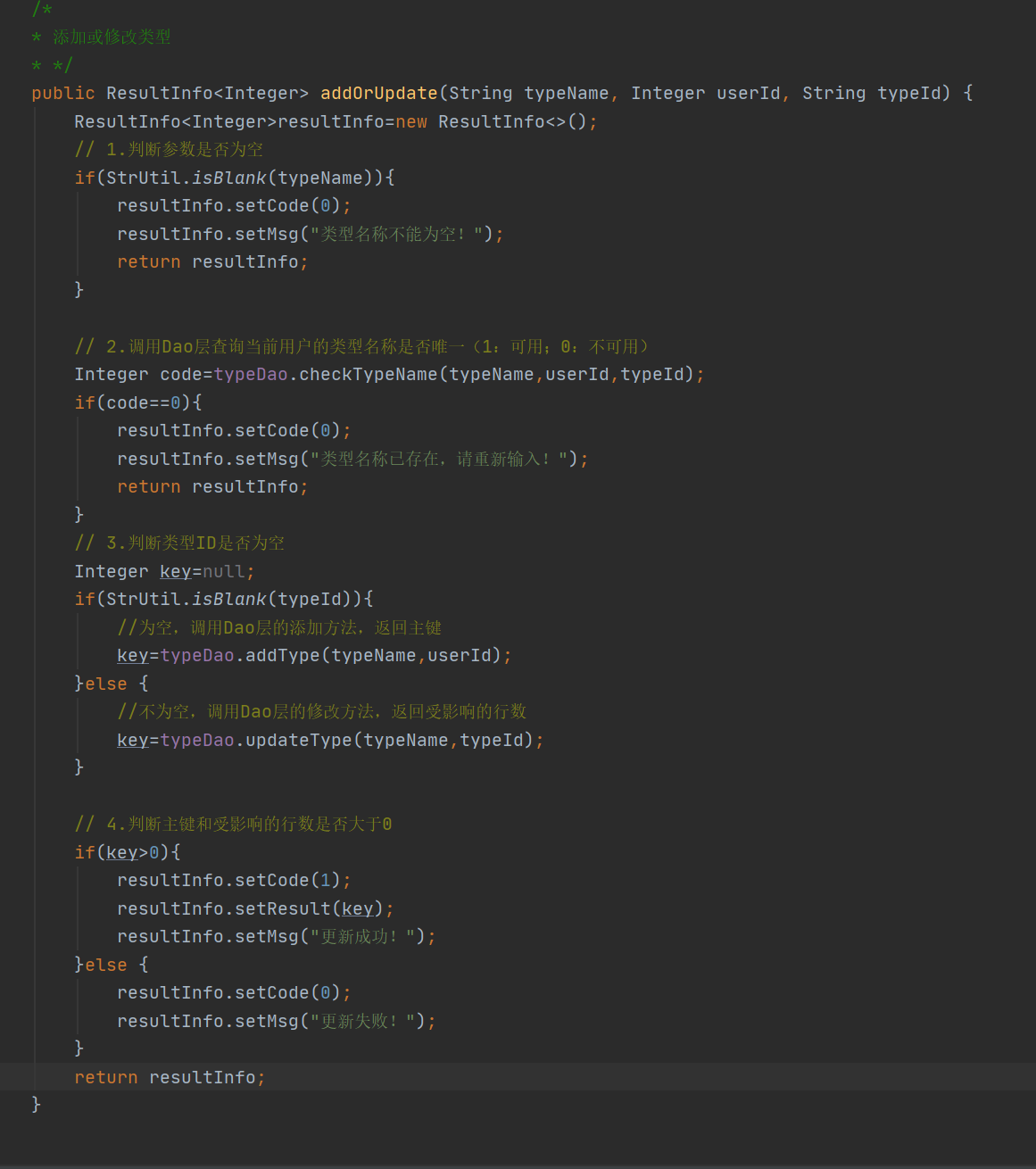


类型模块：

1. 删除某一类型：

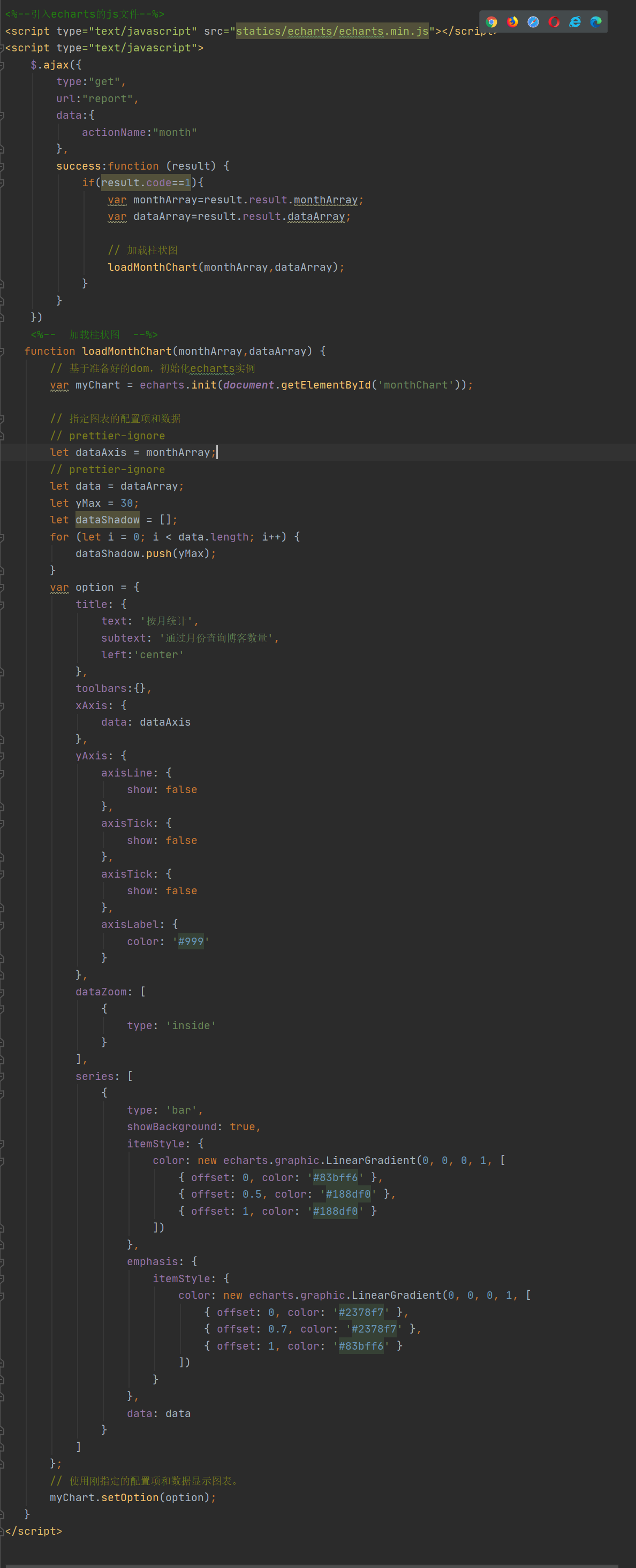


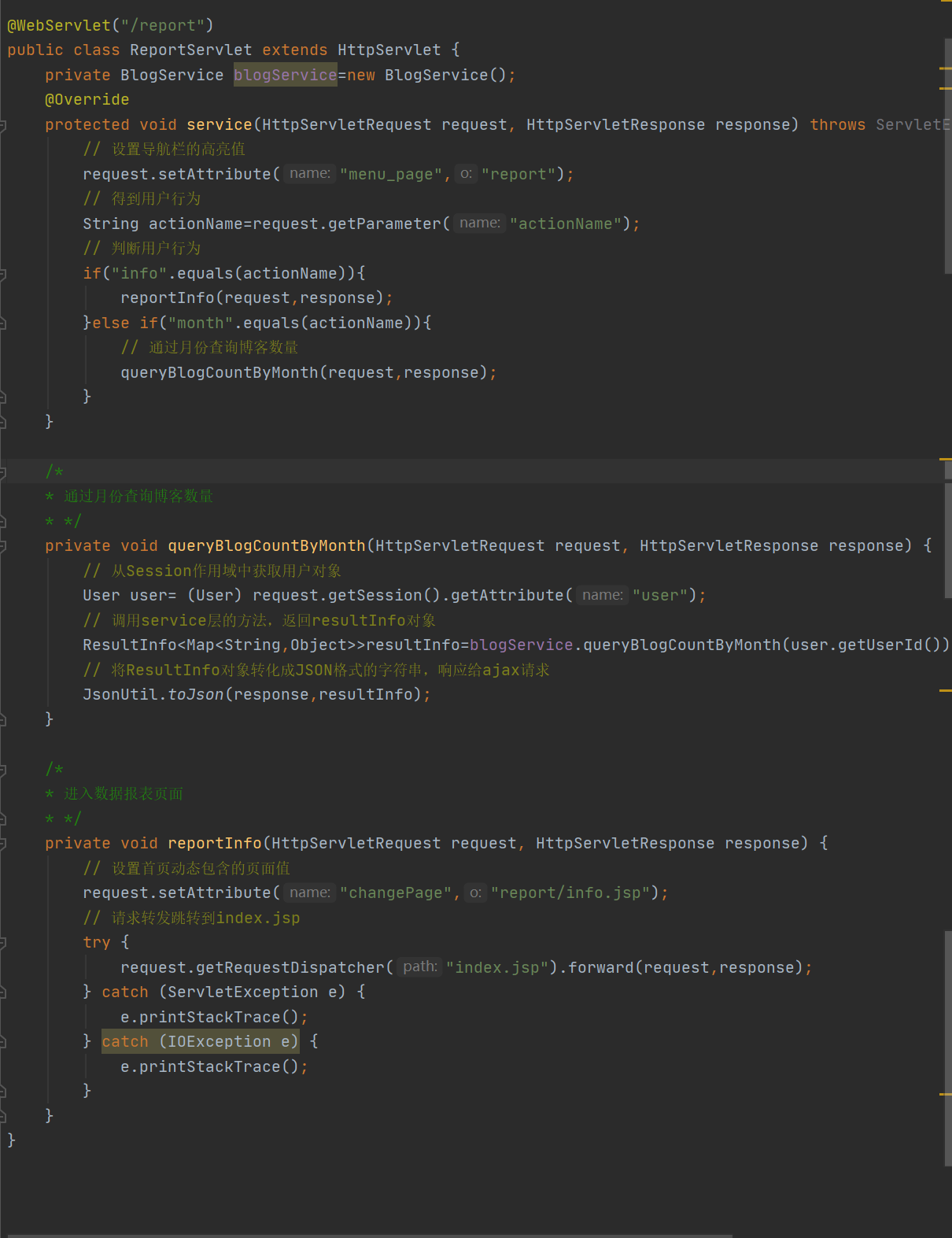
1. 增添或修改类型：



报表统计模块：

该是通过前端页面发送ajax请求，获取每个月份的博客数量，并借助Echarts实现：



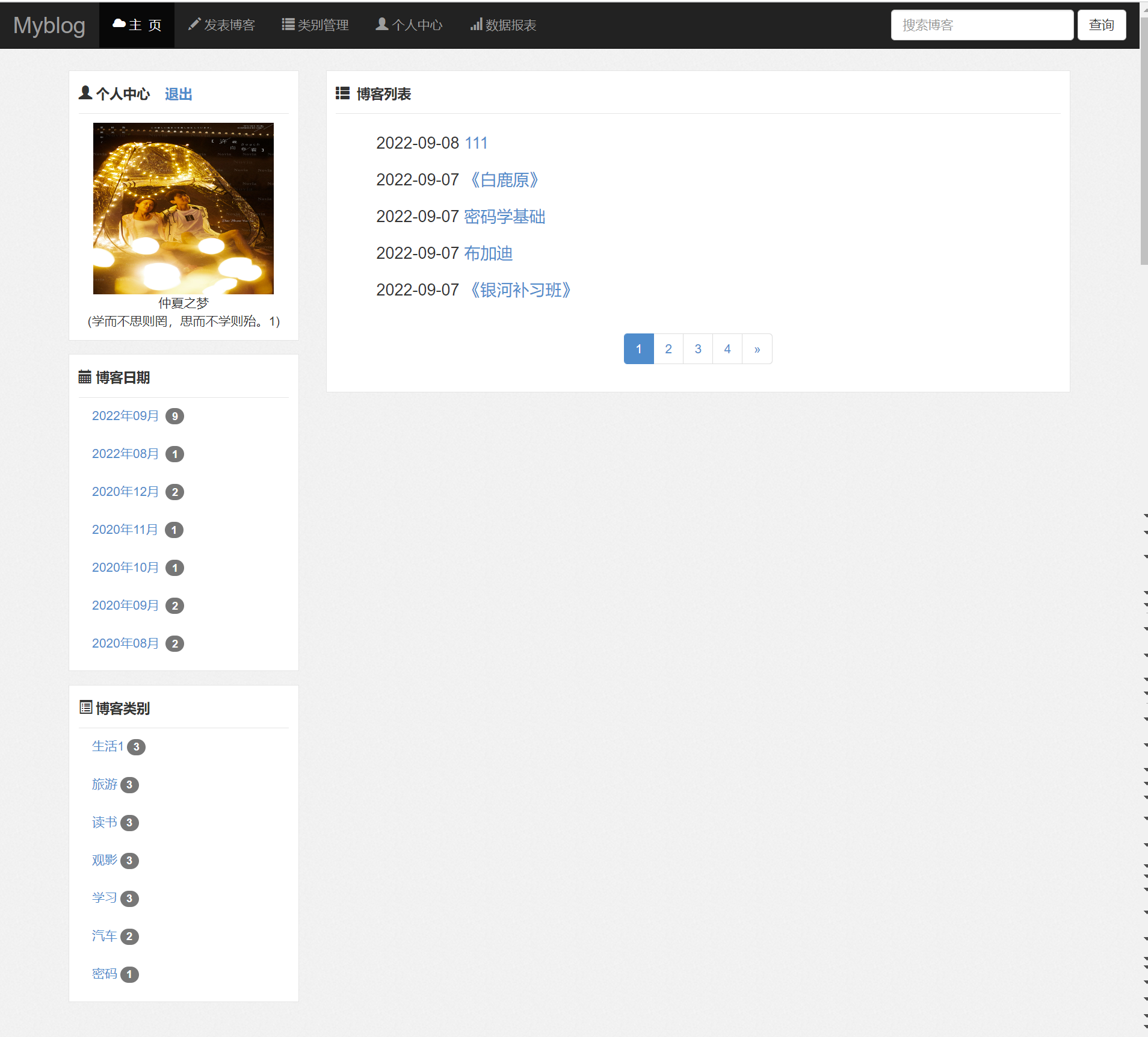


**7.2界面设计：**

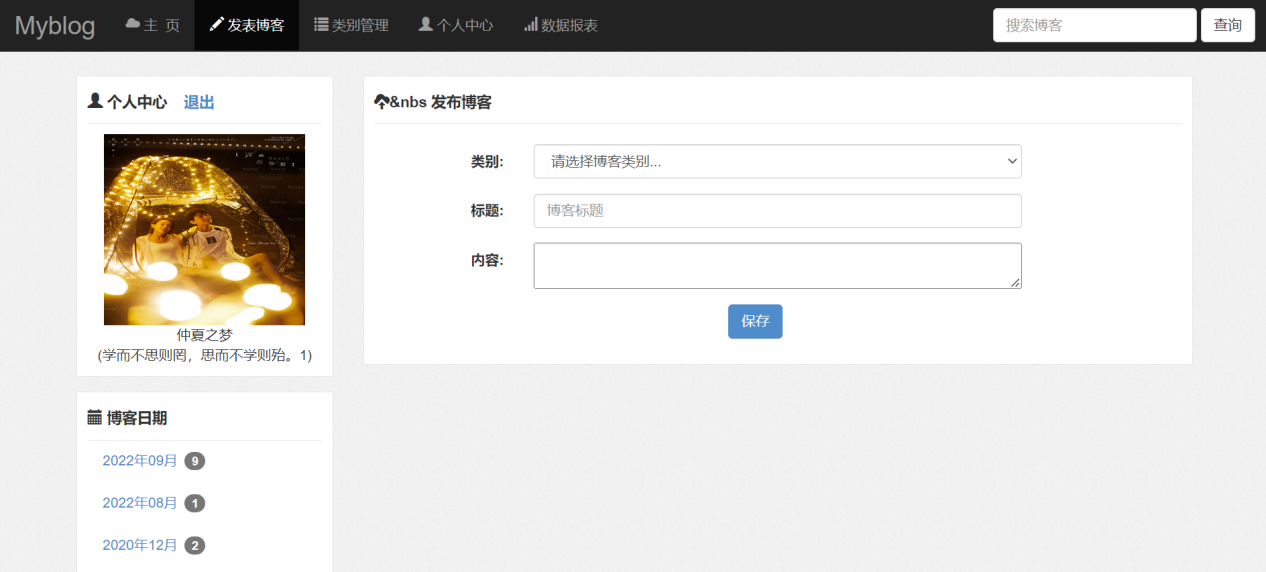
登陆界面：



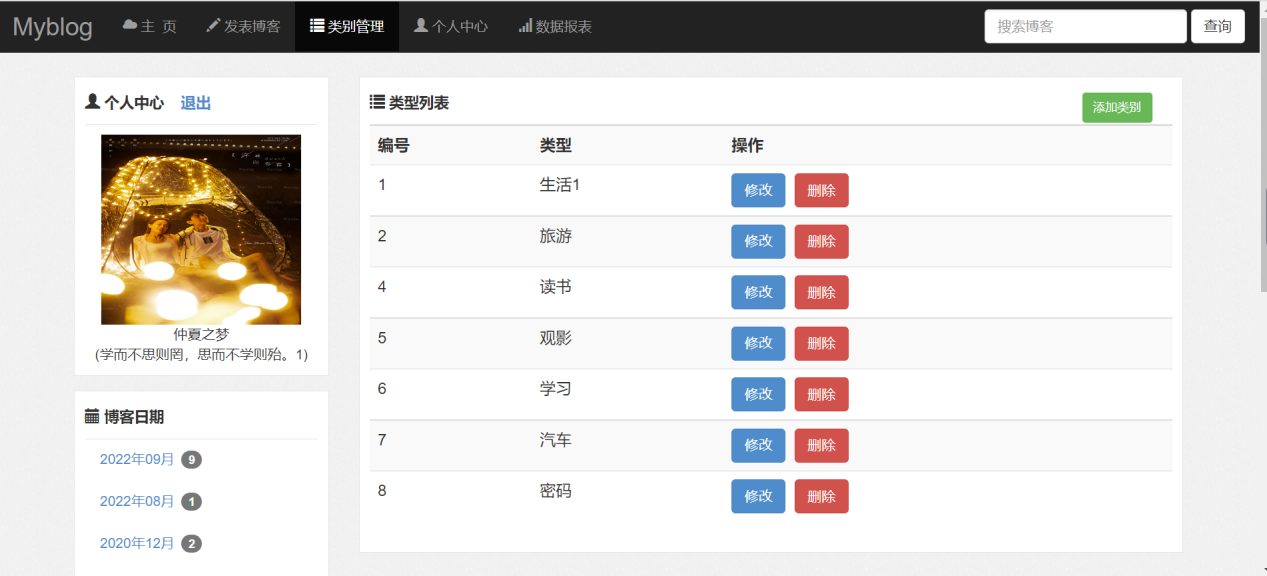
用户主页：



用户发布博客界面：



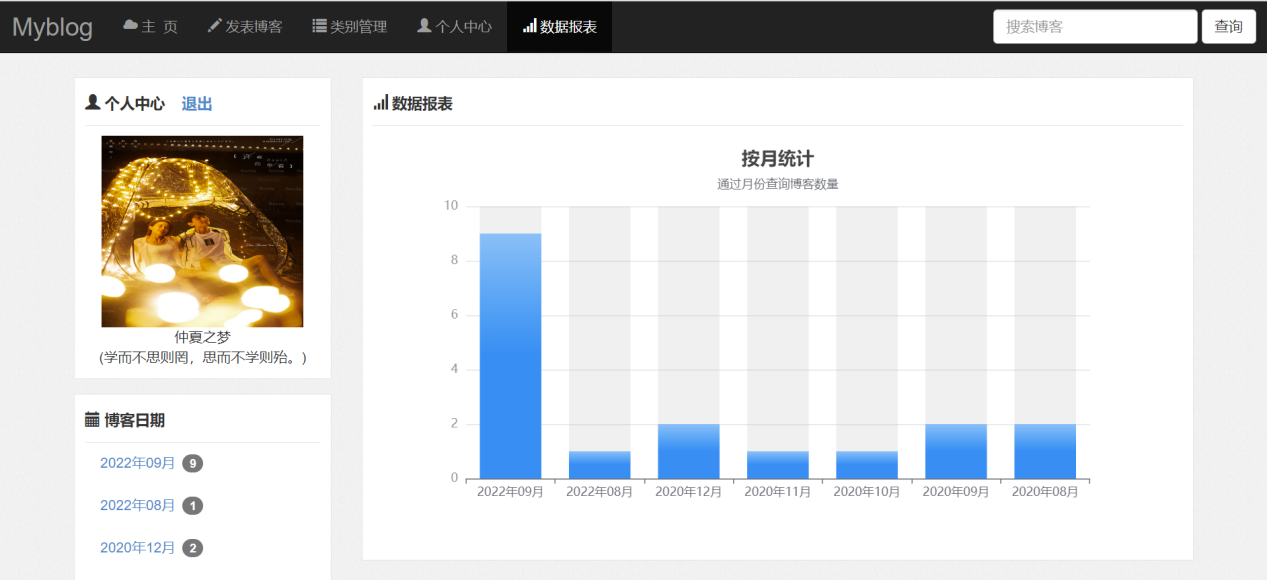
博客类型界面：



用户个人中心界面：



用户数据报表界面：



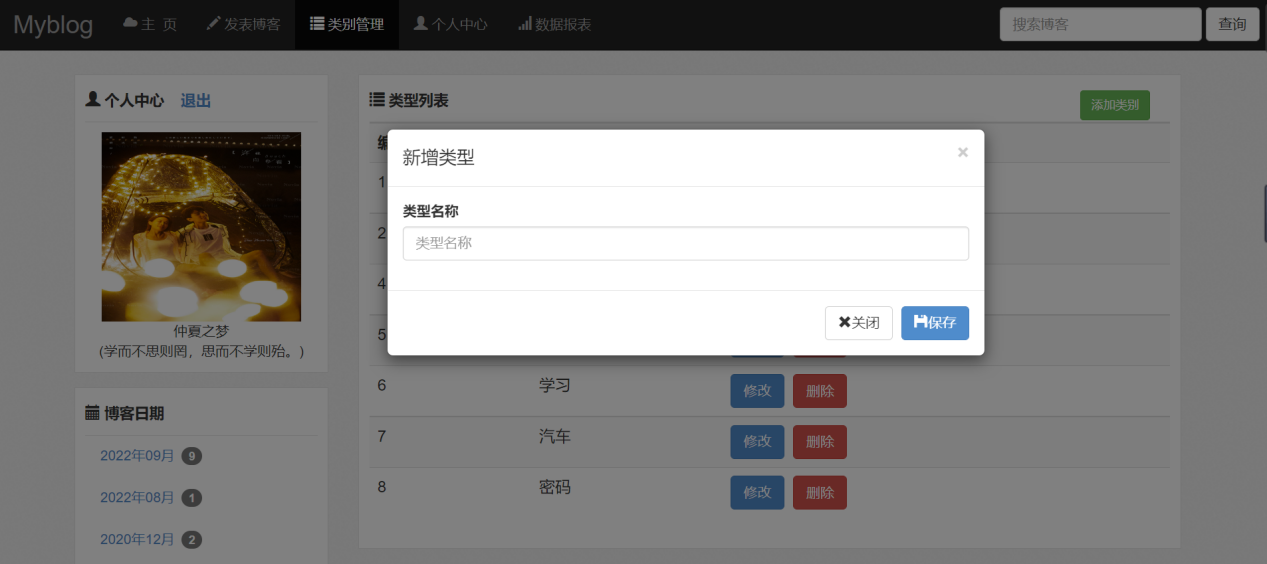
用户博客详情界面：



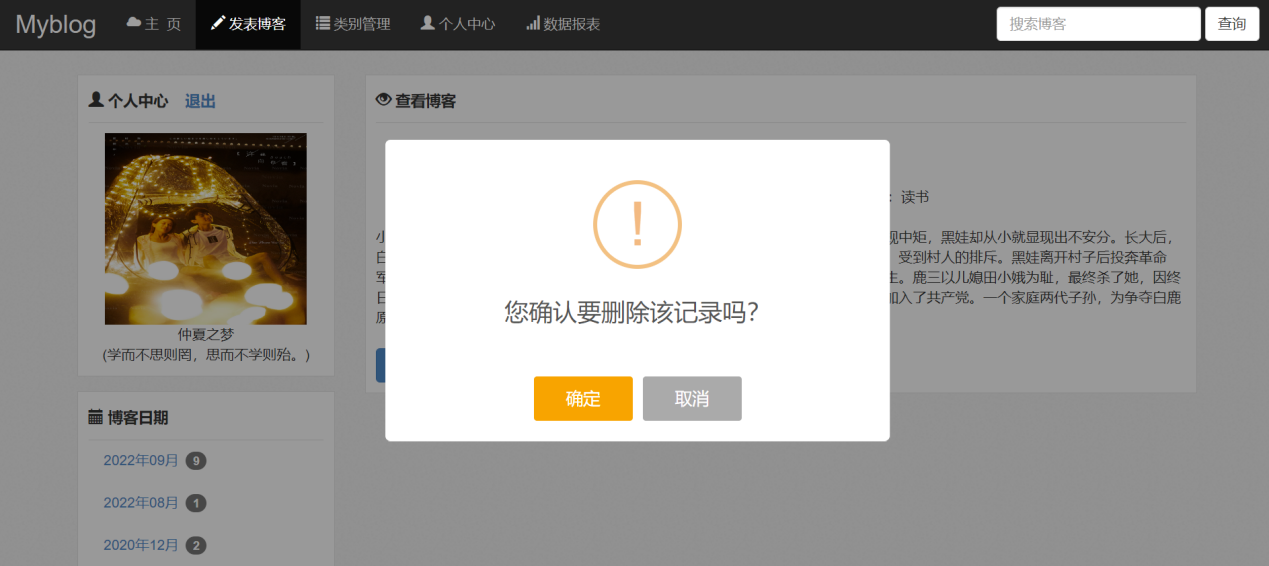
用户类型修改界面：



新增博客类型界面：



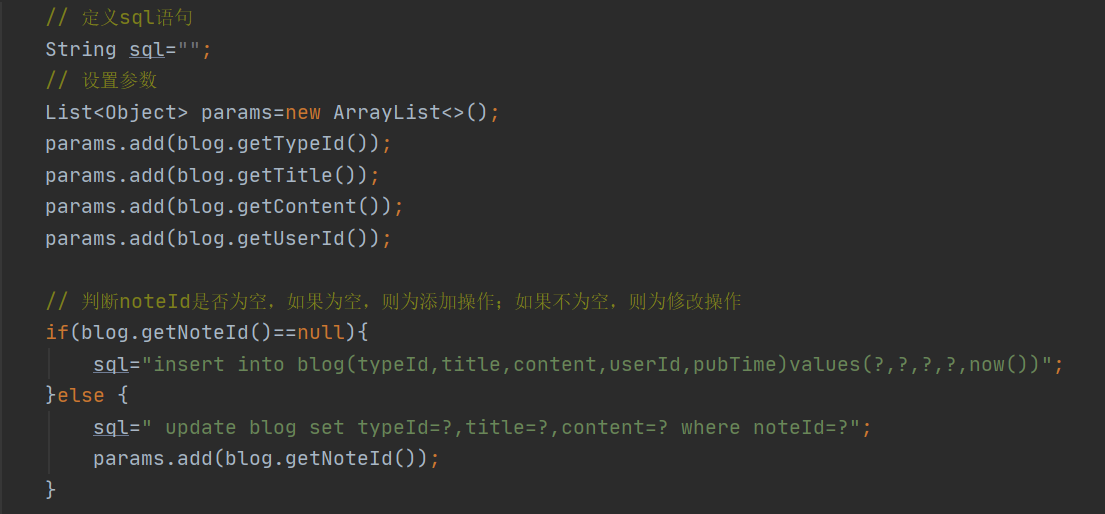
博客删除页面：



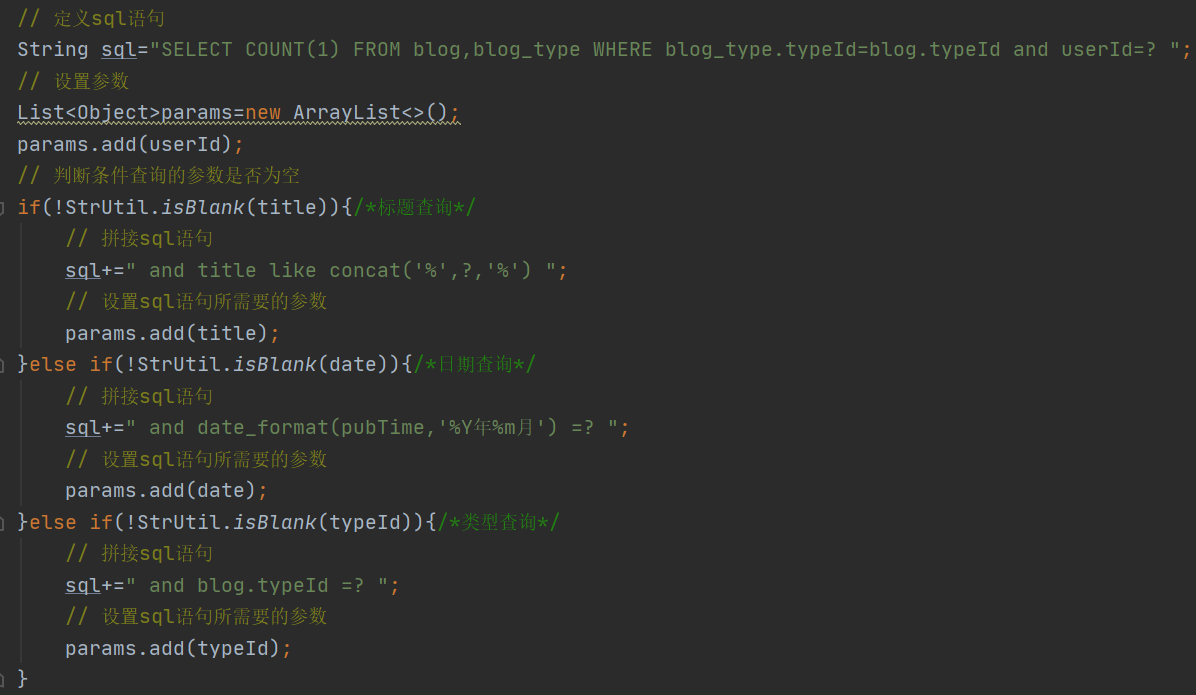
**7.3事务设计：**

该系统中所有的SQL语句都采用预备语句的形式，在很大程度上防止了SQL注入的风险，下面展示系统中关键的SQL语句：

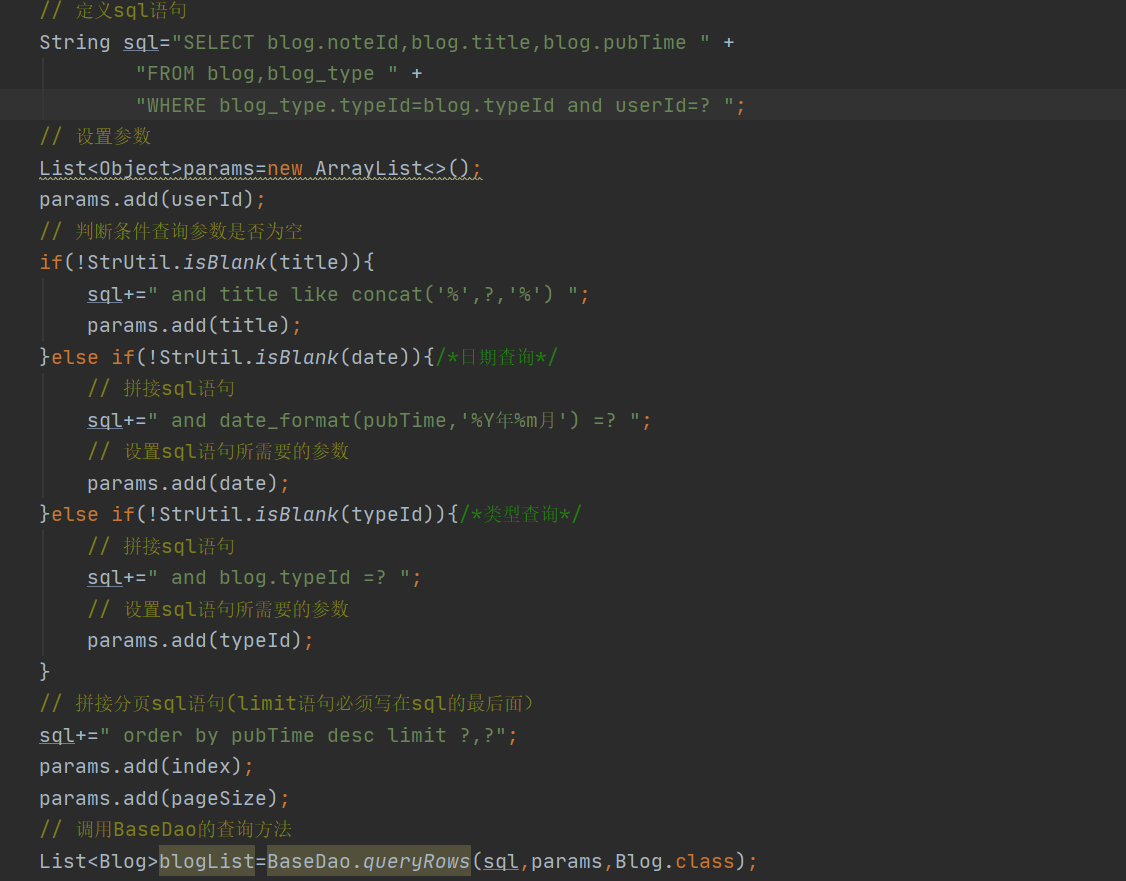
1. 添加或修改博客：



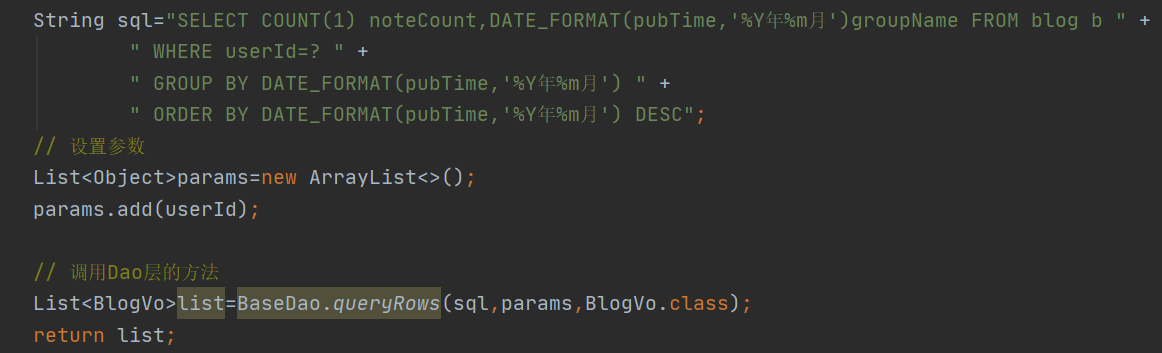
1. 以不同的条件查询博客数量：



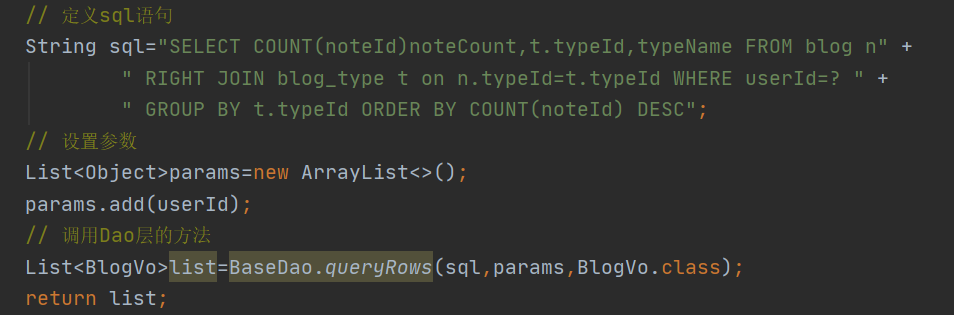
1. 分页查询博客列表：



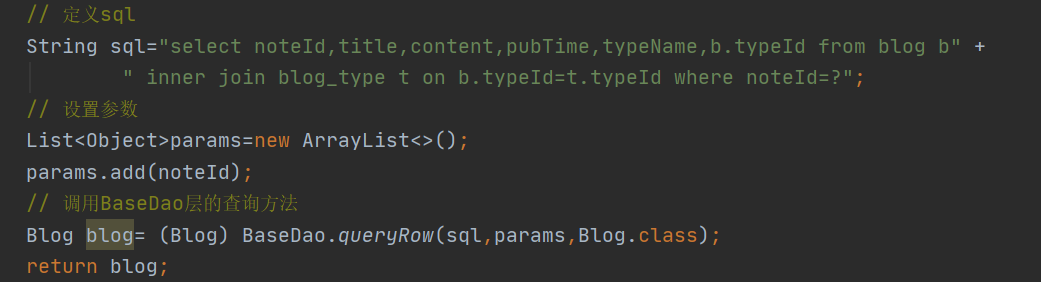
1. 按照日期分组查询博客数量：



1. 通过博客类别查询博客数量：



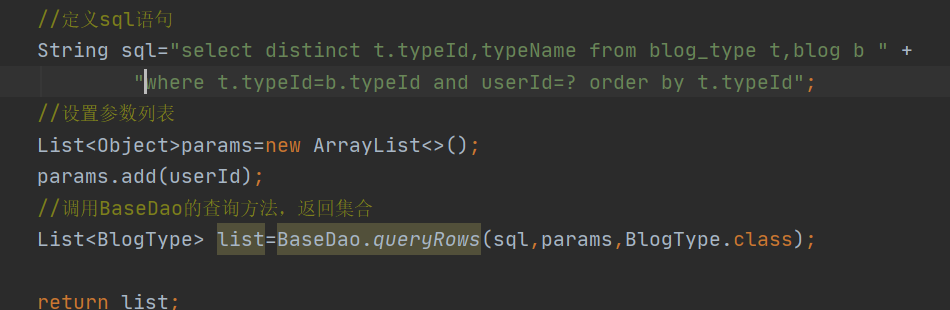
1. 通过Id查询博客：



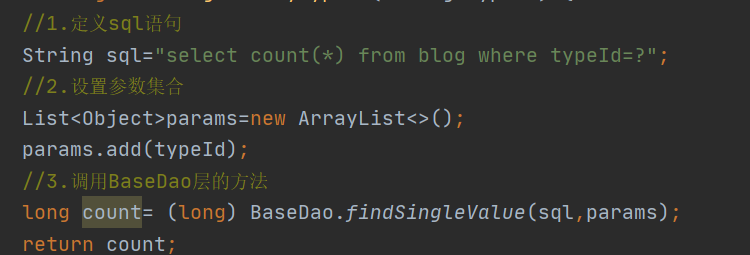
1. 删除博客：



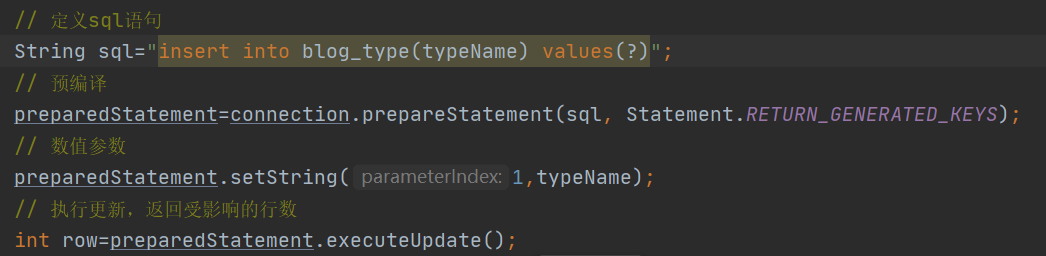
1. 通过用户ID查询类型集合：



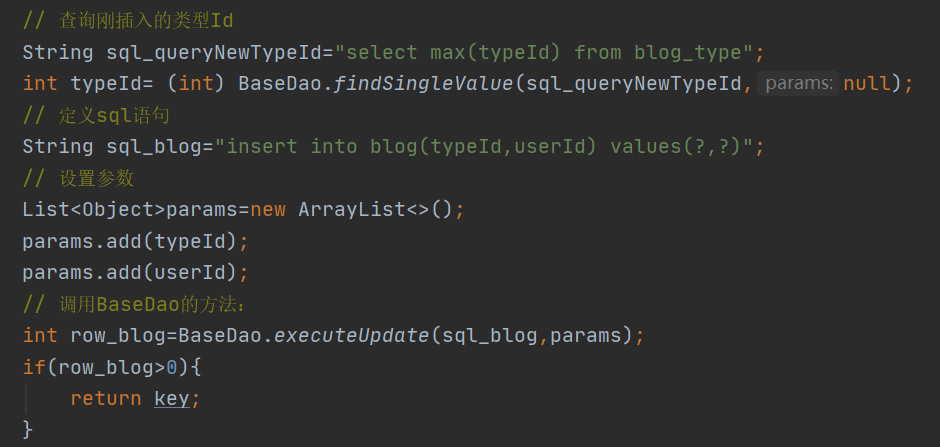
1. 通过typeId查询博客数量:



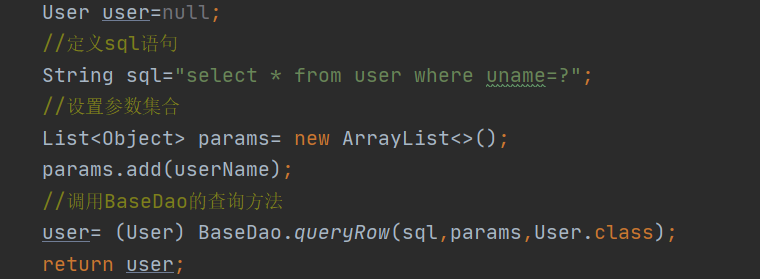
1. 添加博客类型：



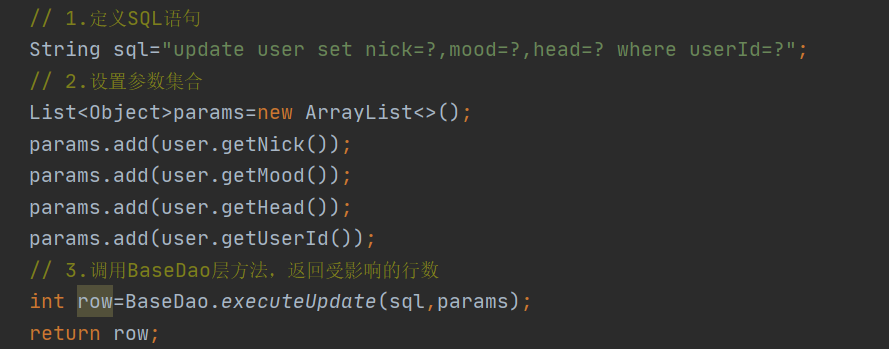
1. 当blog\_type种类型发生改变时，blog表做出相应的改变：



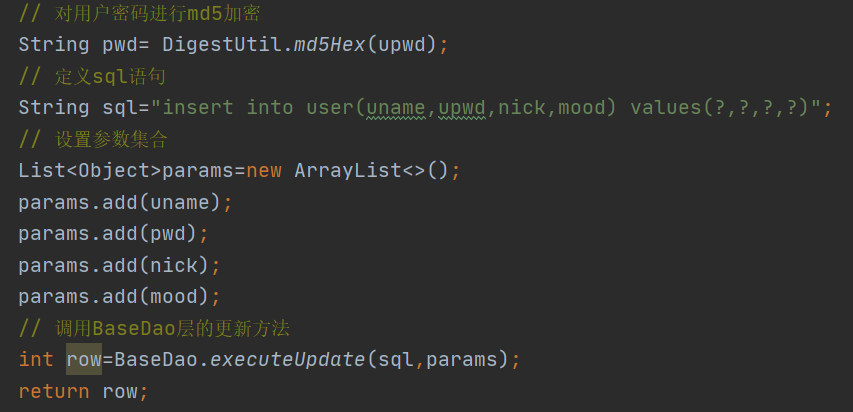
1. 通过用户名查询用户对象：



1. 更新用户信息：



1. 用户注册：

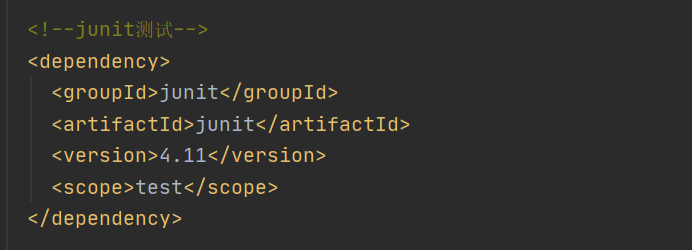


1. **测试和运行**

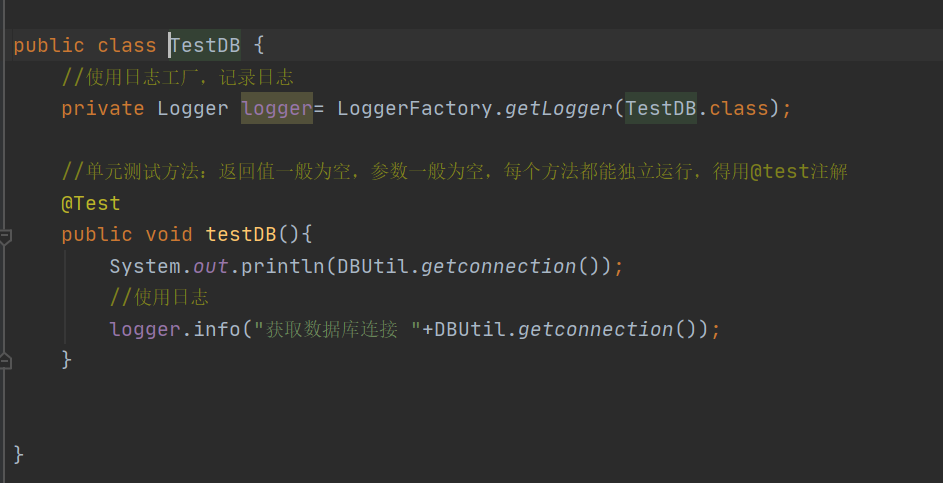
该系统引入了log4j便于系统测试：



该系统引入Junit单元测试框架，方便单元功能测试：



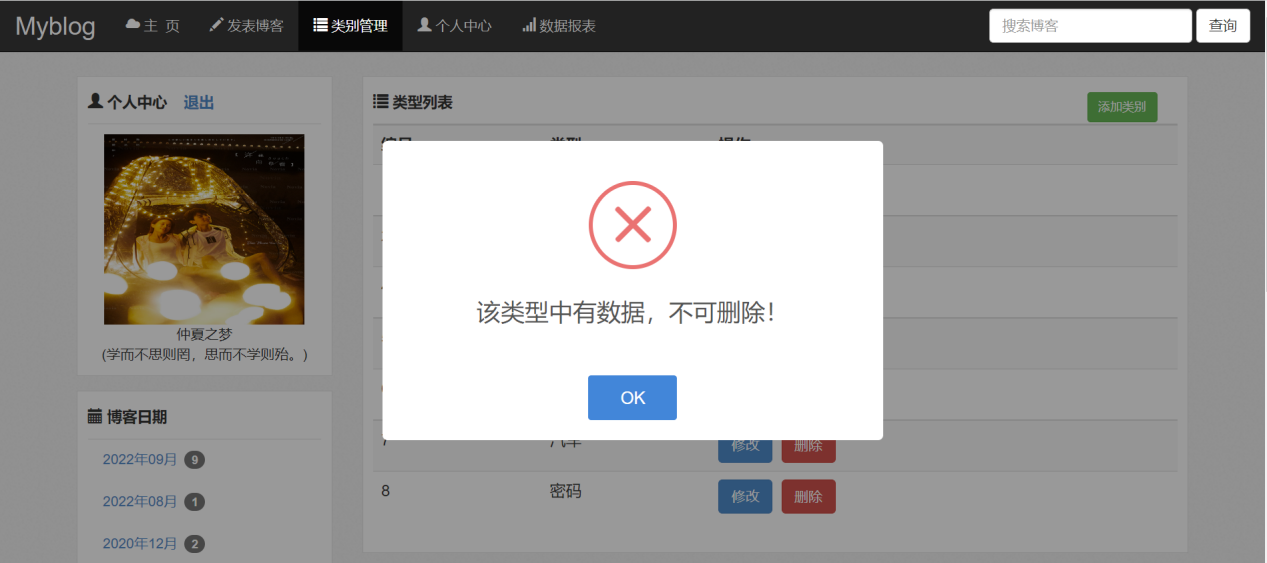
下面是利用该测试框架和log4j日志打印编写的java类，用于测试数据库是否连接成功。



系统登陆测试（当输入错误密码时）：



当某一博客类型中还有博客，则会提醒我们该类型不可删除：



发布博客时，如果不选择博客类型或者博客标题，则会有错误提示：



当所有的信息都填写完整时，则添加成功：



1. **总结**

这个暑假里我前前后后花了一个多月的时间来做数据库课程设计，从前期的资料搜索、项目准备，到中期的数据库设计再到最后的代码编写，这一个月我不敢说自己有多大的进步，获得了多少知识，但起码是了解了项目开发的部分过程理清了项目开发的前端、后端、数据库、服务器这四者的关系。虽说我们也做过java程序设计的课程设计，但前端页面，服务器等都是老师配好了给我们的，那个项目也没有连接数据库，但这次课程设计给我提供了一个很好的机会。从头到尾都是自己一个人精心准备，遇到问题的就上网寻求答案，自己也有一些小骄傲。

通过这次课程设计发现这其中需要的很多知识我们没有接触过，比如Junit单元测试框架、lombok插件的使用使代码更加简介美观等，上网查找资料的时候发现我们以前所学到的仅仅是皮毛，还有很多需要我们掌握的东西我们根本不知道。同时也发现有很多已经学过的东西我们没有理解到位，不能灵活运用于实际，不能很好的用来解决问题，这就需要自己不断的大量的实践，通过不断的自学，不断地发现问题，思考问题，进而解决问题。在这个过程中我们将深刻理解所学知识，同时也可以学到不少很实用的东西。

这次的数据库课程设计，我写的是一个个人博客管理系统。我从系统目标出发，结合实际的需求分析，自己设计系统模块，进行数据库总体设计、代码编写与调试，确实花费了比较长的时间。在这个过程中，总是会出现很多的问题，自己的设想总是不能如愿地实现，不是这儿出现问题，就是那儿有了毛病，随着代码的越写越多，如果某个地方出现纰漏，就可能很长时间找不到问题的根。于是我吸取教训，每次在编写代码之前先去理清思路，把思路写下来，然后照着一步一步往下写，这样就大大降低了出错的可能，这也使得我的代码逻辑清晰，看起来也比较美观。

在进行系统需求分析的时候，我先上网查了一下博客管理功能应有的功能，再从中筛选，结合自己的能力，选择必不可少的功能模块。这也为后续的项目开发奠定了一个很好的基础。在需求分析过程中，我通过上网查资料，结合我的生活经验，根据可行性研究的结果和用户的需要，分析现有情况及问题。在一个多月的时间里，不断地对程序及各模块进行修改、编译、调试、运行，其间遇到很多问题，经过上网查找资料，最终把它解决了。

通过本次课程设计，再次加深了对SQL语言的实际应用，对数据库的创建、修改、删除方法有了更加深入的了解，通过创建表和删除表、更改表，加深了对数据库的基本操作。也通过这次项目，真正地把数据库和实际项目联系起来，而不只是会像上学期一样写写SQL语句。我发现数据库不是想象中的那么简单的，它涉及到的各种实体、属性、数据流向、数据处理等等。很多时候感觉后面的设计根本无法继续，一定要规划好各个实体之间的联系，最好是画出E-R图来实现来理清各个实体之间的联系，方便后续的数据操作。

通过本次课程设计，我觉得不管做什么，始终都要相信自己。不要畏惧，不要怕遇到困难，什么都要去尝试，有些你开始认为很难的事在你尝试之后你可能会发现原来它并没有你你想象的那么难。如果没有自信，没有目标，没有信心就很难把事情做好，当其他人都在迷茫的时候，自己一定要坚信目标。从学习这个专业，到以后从事这方面的工作都需要不断地去学习去实践，这次实践可以给我们敲一个警钟，我们面临毕业，面临择业，需要这些实践经验，在困难面前要勇于尝试，这是这次课程设计给我的最大感想！也通过这次毕业设计的完成，使我对自己所学知识进一步的了解，同时通过实践发现自己的不足，对知识的掌握及运用存在诸多的不完善。在今后的学习中，我也会秉承着认真的态度，尽力做好每一件事。

1. **附：参考文献：**

**网址：**

**<https://juejin.cn/post/6844903557016076302；>**

**<https://www.runoob.com>**

**书籍：**

**《数据库原理及其应用》**

**《项目开发实践》**