



**本科毕业设计（论文）**

**三维虚拟实验网站建设**

**学 院 信息工程学院**

**专 业 信息工程**

**年级班别 2015级（3）班**

**学 号 3115002311**

**学生姓名 邱文捷**

**指导教师 刘东峰**

2019年 6月

**目 录**

1 绪论 1

1.1课题的研究背景 1

1.2课题的实际意义 1

1.3课题的发展趋势 2

2 开发工具和相关技术介绍 3

2.1 Java与Java EE平台 3

2.2 Spring家族 3

2.2.1 Spring框架 3

2.2.2 SpringMVC框架 4

2.2.3 Spring boot框架 5

2.3 Mybatis （持久化框架） 5

2.4 Luence （全文搜索引擎） 6

2.5 Tomcat （Web容器） 6

2.6 Thymeleaf（模板引擎） 7

2.7 Mysql（数据库） 7

2.8 Maven 8

3 网站需求分析 9

3.1 总体需求分析 9

3.1.1 功能分析 9

3.1.2·网站性能需求分析 9

3.1.3 数据安全分析 10

3.2 界面需求分析 10

3.2.1 前台页面 10

3.2.2 后台页面 11

4 网站功能及数据库设计 12

4.1 功能详细设计 12

4.1.1 前台功能详细设计 12

4.1.2 后台功能详细设计 12

4.2 数据库设计 12

4.2.1 数据需求 12

4.2.2 数据库概念设计 13

4.2.3 数据库表设计 14

5 网站具体实现 16

5.1 开发结构 16

5.1.1 网站总开发结构图 16

5.1.2 各部分说明 17

5.1.3 业务逻辑控制流程 21

5.2 前台功能模块实现 21

5.2.1 注册模块 21

5.2.2 登陆模块 23

5.2.3 导航条模块 24

5.2.4 国际化模块 25

5.2.5 搜索框模块 25

5.2.6 首页模块 26

5.2.7 结果页模块 27

5.2.8 详情页模块 28

5.3 用户后台功能模块实现 30

5.3.1 查看实验 30

5.3.2 实验创建 31

5.3.3 图片及实验上传 31

5.4 管理员后台功能模块实现 32

5.4.1 公告和使用案例模块 32

5.4.2 查看用户模块 33

总结 34

参 考 文 献 35

致谢 36

# 1 绪论

## 1.1课题的研究背景

Unity 是一款由Unity Technologies研发的跨平台D /D 游戏引擎[1]。可以用于开发所有的主流的操作系统和平台的各种动画视频或电子游戏，相较于其他大型的游戏引擎如虚幻引擎等，它具有入门门槛低、支持多平台、插件及资源非常丰富、社区成熟等等非常多的优点，现在已经广泛的应用在电脑游戏，收集游戏，VR增强现实以及虚拟实验等等领域。

但是非常可惜的是，现在虚拟实验在国内的应用并不广泛。相关的企业比如说亚太盛世，凭借其在虚拟实验领域的多年深耕，现在已经成长为国内非常知名的教学软件研发和销售公司，但是总体来说这个市场还远远未饱和，仍需要注入些新鲜血液。使用Unity开发虚拟实验软件，依靠着Unity的上述优势，相信必将能够大力推广虚拟实验的应用。

## 1.2课题的实际意义

虚拟实验目前应用前景非常的广泛，它是计算机的图像技术、虚拟现实技术和仿真教学的一种很有效的结合，在一些实验场景和实验设备比较缺乏的学校和地区里，虚拟实验可以提供非常逼真的实验器材和实验材料，然后在一个虚拟的实验环境里进行实验，以期提供较为真实的实验体验，帮助学生学习相关知识。

虚拟实验管理系统对于师生门和实验创作者们的实验管理提供了非常大的便利，收集和整合了教学中遇到的各种类型的实验方向和资源，方便了实验的查询和下载，免去了师生们对于实验资源的漫无目的的搜索。通过该平台，将实验的上传，信息的修改，实验的分类查询，实验的模糊搜索，实验的下载以及后续的评论等各个环节有效的组织在一起，通过一个统一的平台，将所有的关于实验的信息进行管理，使用了该平台，将能够大大的简化了师生们和实验创作者的操作，提高工作效率。

## 1.3课题的发展趋势

可以预见，在未来虚拟实验必然是处于高等学校招生规模在不断的扩大，对于教学上面有各项冲击。其中一点就是对于高校实验以及实验经费上出现不足的情况。实验设备过于陈旧、实验场地也相当拥挤、实物实验次数下降等等情况的出现，对于学生们学习知识有很大影响。

为了保证和提高实验教学的质量，各个科技公司各显神通的进行优化实验设施。采用计算机软件来进行仿真虚拟实验。例如NOBOOK虚拟实验室的系列产品NB物理实验等等软件，利用这些软件在计算机上建立“虚拟实验室”，来虚拟相关的电学、力学等等物理学的实验、元件、仪表等等，组成教学时候的实验，提高教学水平。

发展虚拟实验的目的就是让更多的同学在教室中享受到“身临其境”的实验感觉，直接观察实验产生的现象，了解实验了解相应的原理。因此，大力构建虚拟实验室系统是现代化教育发展的必然趋势。利用高校中的计算机、多媒体及网络环境，让实验成为学习中的重要一环，不用考虑过多的限制，对于学习来说具有实用价值和重要的现实意义。

# 2 开发工具和相关技术介绍

## 2.1 Java与Java EE平台

Java是一种广泛使用的计算机编程语言，拥有跨平台、面向对象、泛型编程的特性，广泛应用于企业级Web应用开发和移动应用开发[2]。Java不同于一般的编译语言或解释型语言，准确的说，它应该属于编译解释型语言。Java文件首先被编译为后缀为.class的字节码文件，然后再经由各种操作系统平台下的虚拟机(JVM)解释执行，因此具有"一次编译,到处运行"的特性。虽然java的这个特性导致的运行效率低下问题饱受诟病，但是自Java SE.版本发布以来，包括Java的创始人James Gosling等人针对底层代码做了许多的优化，以及近些年来计算机cpu的发展迅猛，使得java的运行效率也有了很大提升。Java后来被甲骨文公司收购。

Java当前提供以下三个版本[Java EE](https://zh.wikipedia.org/wiki/J2EE" \o "J2EE)（[Java平台企业版](https://zh.wikipedia.org/wiki/J2EE" \o "J2EE)）、[Java SE](https://zh.wikipedia.org/wiki/J2SE)（[Java平台标准版](https://zh.wikipedia.org/wiki/J2SE" \o "J2SE)）和[Java ME](https://zh.wikipedia.org/wiki/J2ME" \o "J2ME)（[Java平台微型版](https://zh.wikipedia.org/wiki/J2ME" \o "J2ME)）。Java EE，全称是Java Platform Enterprise Edition，2018年3月更名为Jakarta EE[4]。Java EE当前发展非常迅猛，成为[Java](https://zh.wikipedia.org/wiki/Java" \o "Java)语言中最活跃的体系之一，而现在Java EE更多的被人们用来进行设计软件的架构和思想。Java EE是一系列技术标准所组成的平台，包括JDBC、JNDI、EJB、RMI、JSP、Java Servlet、XML、JMS、Java IDL、JTS、JTA、JavaMail和JAF[3]。Java EE技术能够被广泛的使用，是因为许多开源的框架，尤其是以spring家族为首，大大的简化了程序编程的复杂性，给企业的开发带去了许多的便利，例如JDBC给数据库的连接提供了一个方法使得平台数据具有独立性。而JSP技术作为Java的服务器页面，使得Java EE具有可视性，使得企业级应用更具有可操作性。而最重要的servlet的应用，使得Java EE在各个功能上有了很好的改进。

## 2.2 Spring家族

### 2.2.1 Spring框架

Spring框架，即Spring Framework，是一个开源的基于[Java](https://zh.wikipedia.org/wiki/Java" \o "Java)和[Java EE](https://zh.wikipedia.org/wiki/Java_EE" \o "Java EE)全功能栈的[应用程序框架](https://zh.wikipedia.org/wiki/%E8%BD%AF%E4%BB%B6%E6%A1%86%E6%9E%B6" \o "软件框架)，是Spring家族中最核心的框架，该框架最初由Rod Johnson和Juergen Hoeller等开发，最早是出现于[Expert One-on-One Java EE Design and Development](http://www.wrox.com/WileyCDA/WroxTitle/productCd-0764543857.html)一书中的代码[5]。Spring框架为应用的开发提供了一个简易的开发方式，通过xml文件配置方式，避免了底层的代码变得臃肿复杂，Spring包含了如下核心特性：

1、采用控制反转来配置JavaBeans。

2、提供一个核心的Bean工厂。

3、将数据库事务一般化抽象层，简化事务的划分使之与底层无关。

4、灵活的基于核心 Spring 功能的MVC网页应用程序框架。

5、支持使用基于切面编程的AOP框架。

### 2.2.2 SpringMVC框架

MVC设计模式是软件工程里的一种软件设计思想，其核心就是将程序根据其功能分成视图层，控制层和模型层[8]。这三大部分分工明确，互不干预。M（模型）层负责构建数据的模型，经常是一个特定的数据结构，负责与底层数据库打交道。C（控制）层主要负责请求的转发，如字面意义，根据需求实现跳转。V（视图）层则是肉眼可见的部分，负责可视化界面的排布以及渲染。基于MVC的思想，在SpringMVC中，最核心的类就是DispatcherServlet，顾名思义，这是一个负责分发的Servlet,准确的说是负责请求分发的类。除了这个，SpringMVC还为我们准备了视图解析器，国际化支持以及上传下载文件等功能，同时想要使用也非常的简单，仅仅是Controller和RequestMapping两个接口，SpringMVC就能够正常的工作了。Spring MVC的核心框架图如图2-1所示。

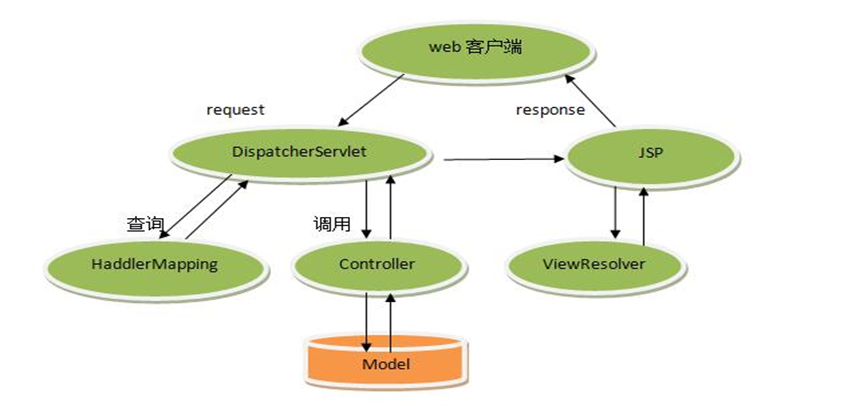


图2-1 Spring MVC的核心结构框图

### 2.2.3 Spring boot框架

如果编写过基于Spring的应用程序应该知道，哪怕我们只是需要实现一个字符串显示功能，也需要大量的配置文件，但是无论是什么项目，这些配置文件都大同小异，程序员们都期望着能从这些繁琐的配置文件中解放出来。于是乎SpringBoot应运而生。SpringBoot框架的主旨就是简洁和轻松，使用SpringBoot,就不再需要写大量的xml配置文件，因为它奉行“约定优于配置”的原则，所有的必须配置项在框架中都有一个默认值，若是没有特殊的需求，完全可以不需要任何配置直接启动一个Spring项目（基于内置的Tomcat容器）。当然要修改默认项也非常的简单，仅仅使用一个Application.properties资源文件或者是在自定义配置类中使用注解等方式就可以完成。如果喜欢Maven，甚至可以在POM文件中更改<dependency>来替换Spring Boot默认值[9]。

## 2.3 Mybatis （持久化框架）

Mybatis是目前非常流行的ORM持久化框架，但并不是一个完整的ORM框架，ORM是对象和关系之间的映射，包括对象->关系和关系->对象两方面。而Hibernate才算是个完整的ORM框架。但是相较于Hibernate而言，Mybatis有着独一无二的优势，那就是自定义sql，从而显得更加的灵活，而且就体量而言也更加的轻量。它主要分为了三个层次。第一层即接口层，核心类为SqlSessionFactory和SqlSession，这几个类是能够实现所有数据库操作的API[7]。简单说就是实现sql语句的增删查改操作。第二层为核心层，主要提供xml的解析操作和SQL操作的具体实现。第三层为基础层，主要提供如日志、缓存、数据连接池、事务以及反射等功能实现。Mybatis的结构层次如图2-2所示。



图2-2 Mybatis的结构层次

## 2.4 Luence （全文搜索引擎）

全文检索技术大多基于倒排索引，所谓倒排索引是相对于正排索引而言，正排索引即通过文档号去搜索文档包括了哪些关键词，而倒排索引是从关键词出发从而找到文档号的方法。Lucene即这么一个全文搜索引擎，它最早是由Doug Cutting个人开发，后来他将其开源，无数的贡献者在其基础上对该引擎进行完善和补充。Lucene为纯java实现，支持多种分词器以及各种查询方式，具有性能高，易扩展的特点。经过了这么多年的发展，Lucene已经成为业内最为成熟的全文搜索框架之一，基于Lucene的Solr和ElasticSearch分布式搜索框架也发展迅速。

## 2.5 Tomcat （Web容器）

Tomcat是由Apache软件基金会属下Jakarta项目开发免费而且开放源代码的Servlet容器，基于Sun Microsystems提供的技术规范的一个小型轻量级的应用服务器[6]。Tomcat的特点是性能高，易扩展，在并发量比较小的网站，是web容器的不二之选。它主要由Server、Connector（连接器）、Engine（引擎）、Host（主机）和Context（正文）组成，在使用的时候，它先启动一个Server，然后通过Connector接收请求，在收到请求后，Engine负责解析请求，并将请求转送到Host，最后Host决定访问哪一个应用的Context。Tomcat的体系结构如图2-3所示。

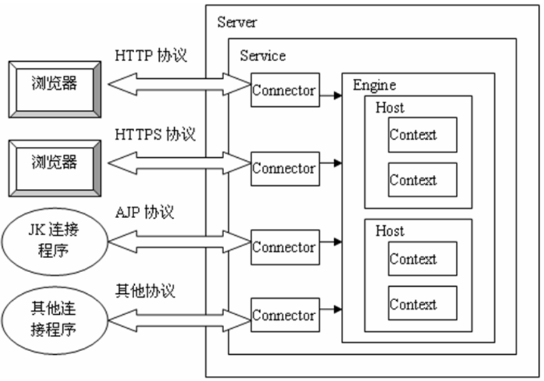


图2-3 Tomcat的体系结构图

## 2.6 Thymeleaf（模板引擎）

Thymeleaf是现在非常流行的模板引擎，它与之前的jsp的最大区别就是不再依赖Servlet，就算是浏览器也能够直接打开文件。由于现在前后端分离已成为大的趋势，故Thymeleaf在使用中更加受到青睐。作为一款模板引擎，它的功能非常丰富，提供多种用途的标签，比如“${}”为变量表达式，“#{}”为国际化表达式，“@{}”为URL表达式等，基本涵盖所有功能[10]。同时，SpringBoot完全支持Thymeleaf，在SpringBoot的官网上也推荐使用Thymeleaf。

## 2.7 Mysql（数据库）

[MySQL](https://baike.baidu.com/item/MySQL/471251" \t "_blank)是一种[开放源代码](https://baike.baidu.com/item/%E5%BC%80%E6%94%BE%E6%BA%90%E4%BB%A3%E7%A0%81/114160" \t "_blank)的关系型[数据库管理](https://baike.baidu.com/item/%E6%95%B0%E6%8D%AE%E5%BA%93%E7%AE%A1%E7%90%86" \t "_blank)系统（RDBMS），使用最常用的数据库管理语言--[结构化查询语言](https://baike.baidu.com/item/%E7%BB%93%E6%9E%84%E5%8C%96%E6%9F%A5%E8%AF%A2%E8%AF%AD%E8%A8%80/10450182" \t "_blank)（SQL）进行数据库管理。

MySQL是开放源代码的，因此任何人都可以在General Public License的许可下下载并根据个性化的[需要](https://baike.baidu.com/item/%E9%9C%80%E8%A6%81/9109492" \t "_blank)对其进行修改。

MySQL因为其速度、可靠性和适应性而备受关注。大多数人都认为在不需要[事务](https://baike.baidu.com/item/%E4%BA%8B%E5%8A%A1" \t "_blank)化处理的情况下，MySQL是管理内容最好的选择。

## 2.8 Maven

[Maven](https://baike.baidu.com/item/Maven" \t "_blank)项目对象模型(POM)，可以通过一小段描述信息来管理项目的构建，报告和文档的[项目管理工具](https://baike.baidu.com/item/%E9%A1%B9%E7%9B%AE%E7%AE%A1%E7%90%86%E5%B7%A5%E5%85%B7/6854630" \t "_blank)软件。

Maven 除了以程序构建能力为特色之外，还提供高级项目管理工具。由于 Maven 的缺省构建规则有较高的可重用性，所以常常用两三行 Maven 构建脚本就可以构建简单的项目。由于 Maven 的面向项目的方法，许多 Apache Jakarta 项目发文时使用 Maven，而且公司项目采用 Maven 的比例在持续增长。

Maven这个单词来自于意第绪语（犹太语），意为知识的积累，最初在Jakata Turbine项目中用来简化构建过程。当时有一些项目（有各自Ant build文件），仅有细微的差别，而JAR文件都由[CVS](https://baike.baidu.com/item/CVS" \t "_blank)来维护。于是希望有一种标准化的方式构建项目，一个清晰的方式定义项目的组成，一个容易的方式发布项目的信息，以及一种简单的方式在多个项目中共享JARs。

## 2.9 layui前端框架

<https://www.layui.com/>

网站的所有后台界面，以及评论打分功能使用到了该框架。

## 2.10 bootstrap前端框架

<http://www.bootcss.com/>

网站的前台布局，以及实验详情页的轮播图使用到了该框架。

# 3 网站需求分析

## 3.1 总体需求分析

### 3.1.1 功能分析

该网站开发的目的是搭建一个有关于unityD引擎在虚拟实验领域的交流使用平台，为应用开发者和广大师生提供一个集查询，搜索，上传和下载于一体的网站，主要的功能有：（1）用户的登陆注册功能，用户类型主要分为普通用户和开发者两种类型。（2）实验的创建和实验信息的修改功能，是开发者的主要操作之一。（3）实验的上传功能，包括实验本体上传以及实验介绍图片的上传。（4）实验的查询功能，可以通过实验的分类，实验标题，关键词和实验的描述模糊查询实验。（5）国际化功能，为网站配置了两种语言环境，可以实验随时切换。（6）评论功能，已登陆用户可以对实验进行评论和打分操作。（7）管理员功能，编辑使用案例和公告，查看所有用户信息与实验信息。

详细的功能需求如表3-1所示

**表3-1功能需求一览表**

|  |  |  |
| --- | --- | --- |
| 业务模块 | 功能划分 | 简要描述 |
| 登陆注册 | 登陆 | 根据用户邮箱和密码登陆 |
| 注册 | 填好个人的信息后注册成为用户 |
| 实验模块 | 创建 | 实验相关信息填写 |
| 修改 | 实验相关信息修改 |
| 上传 | 图片 | 实验详情页展示的图片 |
| 实验 | 实验详情页下载的实验 |
| 查询 | 实验查询 | 通过标题，关键词，实验描述查询 |
| 评论 | 实验评论 | 为实验评论和打分 |
| 国际化 | 全局国际化 | 随时切换不同的语言环境 |
| 管理员模块 | 使用案例和公告 | 编辑网站公告和使用案例 |
| 用户及实验 | 查看所有已注册用户及实验信息 |

### 3.1.2·网站性能需求分析

本网站主要面向的对象是在校的师生和虚拟实验的开发者，并非是商业性的网站，因此并发量不会很高，故性能需求如下：

1、因为并发量不高，对于web服务器选择而言，springboot内置的Tomcat就能满足。

2、网站页面应做到内容全面且直观，操作要简单容易上手

### 3.1.3 数据安全分析

因为网站是一个中间平台，必然会涉及到个人信息的安全问题，信息如果泄露了，将会造成非常严重的后果，因此网站必须确保个人信息的安全，以及能够拥有拒绝非法请求的能力，数据安全主要有以下三点：

1、每一个用户在网站都会有一个唯一的标识，在网站的数据库中被唯一的保存着，在用户进行注册或者登陆操作时，网站会对其身份进行认证，只有网站核实了个人信息之后，才能够进入网站系统，尤其是在注册操作时，还需要核验验证码，防止恶意注册影响网站运作。

2、网站主要存在两种用户，对于不同的用户而言，他们拥有不同的权限，网站需要在视图和后台两方面进行区分。首先是视图方面，用户只能够看见自己所拥有的权限，与自己无关的权限视图时不可见的，其次是后台方面，在请求一个和用户无关的链接时，网站应予以拦截，通过这两方面，确保了严格的权限划分。

3、用户的登陆密码在后台中使用MD消息摘要加密，它使用单项散列函数算法，它一般用于验证信息的完整性，此处我们用来对密码进行加密，另外还会多采用一个“盐”值，确保了数据库即使泄露，也能够保证密码的安全。

## 3.2 界面需求分析

网站主要分为前台与后台界面，此外还包括有登陆和注册页面。

### 3.2.1 前台页面

1、前台页面主要以黑色为基调，包括了首页，搜索结果页和详情页。主布局界面如图3-1所示。

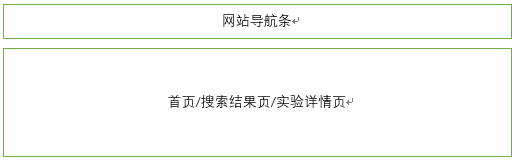


图3-1 网站前台主布局

2、最上方的导航条主要显示网站名称，提供注册及登陆、切换语言环境、进入后台以及提供搜索框的功能。

3、中间根据不同的情况显示不同的界面，首页提供三种分类的实验链接；搜索结果页主要展示各种方式的实验搜索结果；实验详情页则提供实验图片介绍，实验文字简介，实验评价，实验下载链接等功能。

### 3.2.2 后台页面

1、网站后台主布局如图3-2所示。

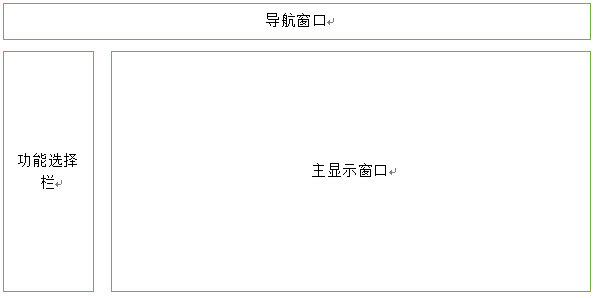


图3-2 网站后台主布局

2、导航窗口提供回到首页，展示个人信息，退出登陆等功能。

3、功能选择栏主要为实验服务，包括创建删除实验，上传实验等功能。

4、主显示窗口则为功能选择栏提供显示具体操作的显示窗口。

# 4 网站功能及数据库设计

## 4.1 功能详细设计

### 4.1.1 前台功能详细设计

前台是直接与用户进行交互的界面，主要的功能设计如下：

1、提供一个一直存在的导航条，能够实现快速回到首页的功能，能够显示登陆的状态。

2、导航条提供登陆和注册功能，跳转到登陆和注册界面，显示用户的登陆状态，同时导航条还提供一个搜索框，可以根据关键词进行搜索。

3、首页根据根据实验分类进行查找实验，显示网站公告和使用案例

4、实验详情页提供实验下载，实验评论，实验详情查看功能。

### 4.1.2 后台功能详细设计

后台是实验的操作界面，主要的功能设计如下：

1、实验的创建、删除、修改。

2、个人信息的查看、修改。

3、实验评论信息的查看。

4、个人信息的查看、修改。

5、公告和使用案例的增加与删除

## 4.2 数据库设计

### 4.2.1 数据需求

网站主要有用户和实验两大元素，所以经过对它们的分析之后，总结出下面的内容：

1、记录用户信息需要有一个用户信息表。

2、记录实验信息需要有一个实验信息表。

3、记录实验的评论信息需要有一个实验评论记录表。

4、保存网站的公告需要有一个网站公告表。

5、保存使用案例需要有一个实验使用案例表。

### 4.2.2 数据库概念设计

1、网站的数据流图

网站的数据流包括了用户和实验信息及其他的记录信息，数据的流动如图4-1所示：

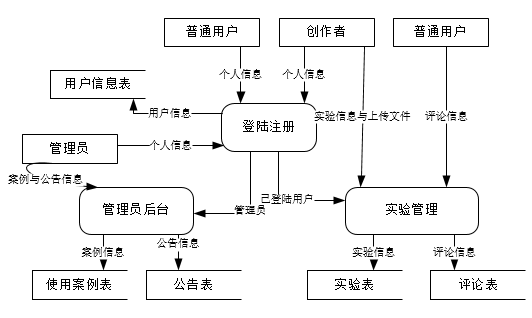


图4-1 数据流图

2、用户实体

用户实体拥有的属性为用户编号，用户昵称，用户邮箱，登陆密码，用户类型，注册日期，用户电话。

3、实验实体

实验实体拥有的属性为实验编号，实验类型，实验名称，实验创建者编号，实验简介，实验关键词，实验保存地址，实验获得的打分，实验是否为免费。

4、评论实体。

评价实体拥有的属性为评论编号，评价人编号，评价人昵称，实验编号，评价内容，评价打分，评价时间。

5、公告实体

公告实体拥有的属性为公告编号，公告标题，公告内容和公告时间。

6、使用案例实体

使用案例实体拥有的属性为案例编号，案例标题，案例内容，案例图片。

### 4.2.3 数据库表设计

本网站数据库表名为unity，总共有张表，包括了用户表，实验表，实验评论表，网站公告表，使用案例表。

1. 用户表，如表4-1所示

**表4-1 user表**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 序号 | 字段名 | 数据类型 | 长度 | 允许为空 | 是否为主键 | 说明 |
| 1 | id | Int | 11 | 否 | 是 | 编号 |
| 2 | username | varchar | 20 | 否 | 否 | 昵称 |
| 3 | email | varchar | 40 | 否 | 否 | 邮箱 |
| 4 | tel | varchar | 20 | 是 | 否 | 电话 |
| 5 | regitime | varchar | 40 | 否 | 否 | 注册时间 |
| 6 | usertype | int | 11 | 否 | 否 | 类型 |
| 7 | password | varchar | 20 | 否 | 否 | 密码 |

1. 实验表，如表4-2所示

**表4-2 work表**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 序号 | 字段名 | 数据类型 | 长度 | 允许为空 | 是否为主键 | 说明 |
| 1 | id | int | 11 | 否 | 是 | 实验编号 |
| 2 | workname | varchar | 50 | 否 | 否 | 实验名称 |
| 3 | workmanid | int | 11 | 否 | 否 | 创建者编号 |
| 4 | content | varchar | 500 | 否 | 否 | 实验简介 |
| 5 | pictures | varchar | 1000 | 是 | 否 | 展示图片 |
| 6 | keyword | varchar | 50 | 否 | 否 | 关键词 |
| 7 | worktype | int | 3 | 否 | 否 | 实验类型 |
| 8 | score | int | 10 | 是 | 否 | 获得打分 |
| 9 | workaddress | varchar | 200 | 是 | 否 | 下载地址 |
| 10 | forfree | int | 3 | 否 | 否 | 是否免费 |

1. 实验评论表，如表4-3所示

**表4-3 assess表**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 序号 | 字段名 | 数据类型 | 长度 | 允许为空 | 是否为主键 | 说明 |
| 1 | id | int | 11 | 否 | 是 | 评论编号 |
| 2 | valuerid | int | 11 | 否 | 否 | 评论者编号 |
| 3 | workid | int | 11 | 否 | 否 | 实验编号 |
| 4 | content | varchar | 500 | 否 | 否 | 评论内容 |
| 5 | star | int | 11 | 否 | 否 | 打分 |
| 6 | assessdate | varchar | 40 | 否 | 否 | 评论日期 |
| 7 | assessername | varchar | 40 | 否 | 否 | 评论者昵称 |

1. 网站公告表，如表4-4所示

**表4-4 news表**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 序号 | 字段名 | 数据类型 | 长度 | 允许为空 | 是否为主键 | 说明 |
| 1 | id | int | 11 | 否 | 是 | 公告编号 |
| 2 | newsname | varchar | 100 | 否 | 否 | 标题 |
| 3 | content | varchar | 500 | 否 | 否 | 内容 |
| 4 | ctime | varchar | 20 | 否 | 否 | 创建日期 |

1. 使用案例表，如表4-5所示

**表4-5 usecases表**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 序号 | 字段名 | 数据类型 | 长度 | 允许为空 | 是否为主键 | 说明 |
| 1 | id | int | 11 | 否 | 是 | 编号 |
| 2 | casename | varchar | 100 | 否 | 否 | 标题 |
| 3 | content | varchar | 500 | 否 | 否 | 介绍 |
| 4 | image | varchar | 50 | 否 | 否 | 图片 |

# 5 网站具体实现

本网站主要使用了Maven构建项目，采用Springboot框架搭建，前端页面模板使用Themleaf，服务器使用Tomcat，采用MVC开发模式，网站主要分为前台和后台两部分，主要的功能就是围绕着虚拟实验的创建、查找、下载和评论。

## 5.1 开发结构

### 5.1.1 网站总开发结构图

具体的实现过程结构如图5-1所示：

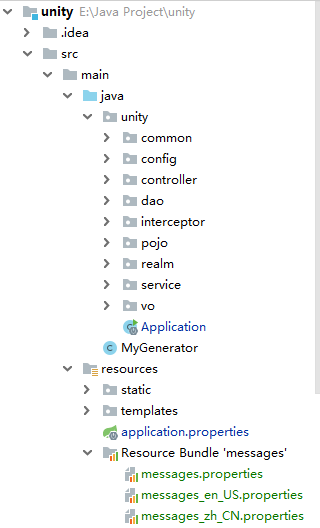


图5-1 网站实现过程结构图

### 5.1.2 各部分说明

1、后端代码

后端代码统一放在unity包里，其中几个比较重要的文件夹，如controller包里存放着控制器代码，负责接收请求并调用相关处理方法；dao和service包分别负责底层数据库操作和业务处理方法；config包里存放配置类文件，例如SpringMVC和Shiro框架以及分页插件等的配置类文件；realm包保存有Shrio框架的Realm域；pojo包里保存有数据库对应的实体类；vo包里则存放着用于页面展示用的实体类；Application类为整个项目的启动类。

1. **controller包**



AdminController：实验创作者的Controller，负责接收的请求有创建实验、上传实验、上传图片等操作,具体参考下文。

AssessController：评论Controller，负责评论请求。

IndexController： 首页的Contrller，负责显示公告和实验案例、跳转后台请求。

LangController： 语言切换（国际化）的Controller，只负责语言切换请求。

RootController： 管理员的Controller，负责管理员的所有操作请求，具体参考下文。

UnityExceptionHandler：全局异常的Controller，负责所有Contrller的异常处理。

UserController：所有用户Contrller，负责登陆和注册请求

WorkController：实验的Controller，负责前台所有和实验有关的请求，有根据分类查询实验，搜索框搜索实验，获取实验详情。

1. **common包**

只有一个自定义的Result类，主要用在layui框架的表格处，详细作用参见下文，主要属性有

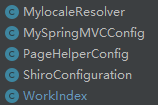
**private** **int** code;       //状态码

**private** **int** count;      //数据的数目

**private** Object data;    //数据

**private** String msg;  //提示信息

1. **config包-配置类**



MylocaleResolver: 语言切换bean，在Application启动类中注入

MySpringMVCConfig: SpringMVC配置类，负责配置拦截器、配置静态资源路径

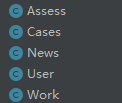
PageHelperConfig: 分页操作的配置类

ShiroConfiguration: Shiro权限控制配置类

WorkIndex: Lucene全文检索配置类

1. **pojo包**

pojo包对应着数据库五个表的实体类，属性即表的字段，分别为



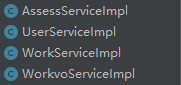
1. **dao包**

dao包主要为数据库底层操作接口，分为java接口文件和xml配置文件，主要是数据库表的增删改查操作。

1. **realm包**

realm包里的Myrealm.java是shrio框架的数据源，负责身份验证和权限认证问题

1. **service 包**



AssessServiceImpl:具体负责评论功能的方法，包括获取某个实验的评论和创建评论。

UserServiceImpl:具体负责用户相关方法，包括获取网站所有用户，通过id获取某个特定用户，通过邮箱查询用户，创建用户

WorkServiceImpl: 具体负责实验相关方法，包括通过id查找特定实验、查询所有实验、查询某个创作者的所有实验等。

WorkvoServiceImpl: 实验的vo类的转换方法类，就是用于展示用的类（pojo的实验实体work类是用来和数据库打交道），这个类负责把work类转换为workvo类

1. **vo包**

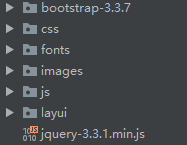


Workvo: 用于实际展示的类，下文会详解

2、资源文件夹

项目用到的所有资源都保存在resources文件夹内，其中static文件夹保存的是静态资源，例如css、js文件，以及网页会用到的一些图片；templates文件夹保存着所有的html网页文件；application.properties文件则是springboot项目的配置文件，配置项包括了端口号，数据库账号密码，日志输出格式和实验保存地址等等。

1. **static**



bootstrap-3.3.7: Bootstrap框架所需的必要文件

css: 所有html页面中需要的css文件

fonts: html页面中的字体文件

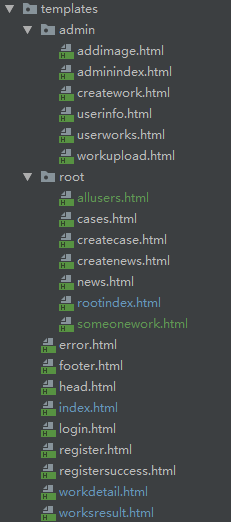
images: html页面中用到的图片

js: html页面中用到的js文件

layui: layui框架所需的必要文件

jquery-3.3.1.min.js: Jquery框架所需要的文件

1. **template**



本网站用到的所有html页面，最下方的error.html到workrsresult.html的9个文件为前台部分的网页，admin文件夹下的所有文件为创作者的后台页面，root文件夹下的所有文件为管理员的后台的页面。

1. **application.properties文件**

**#代表注释，每一项的具体解释如下,而在具体使用时，数据库相关，以及自定义的配置处需要根据具体情况修改。**

1. ###################SpringBoot配置开始#############################
3. #jsp的视图重定向（建议不使用jsp）
4. #spring.mvc.view.prefix=/WEB-INF/jsp/
5. #spring.mvc.view.suffix=.jsp
7. #thymeleaf start
8. spring.thymeleaf.suffix=.html
9. spring.thymeleaf.prefix=classpath:/templates/
10. spring.thymeleaf.mode=HTML5
11. spring.thymeleaf.encoding=UTF-8
12. spring.thymeleaf.servlet.content-type=text/html
13. #开发时关闭缓存,不然没法看到实时页面
14. spring.thymeleaf.cache=false
15. #thymeleaf end

18. #端口号
19. server.port=8088
20. server.tomcat.max-http-header-size=8192
21. #上下文路径
22. #server.context-path=/test

25. #多配置文件选择
26. #spring.profiles.active=pro  pro就是application-pro.properties文件
28. #数据库相关
29. #数据库的连接
30. spring.datasource.url=jdbc:mysql://127.0.0.1:3306/unity?characterEncoding=UTF-8&useSSL=false
31. #数据库的用户名
32. spring.datasource.username=root
33. #数据库的密码
34. spring.datasource.password=123
35. spring.datasource.driver-class-name=com.mysql.jdbc.Driver
36. #自动更新表结构
37. #spring.jpa.properties.hibernate.hbm2ddl.auto=update

40. #上传文件大小设置
41. spring.servlet.multipart.max-file-size=5102MB
42. spring.servlet.multipart.max-request-size=5102MB
44. #国际化文件名及所在位置（i18n是resources下文件夹）,默认值为messages
45. #spring.messages.basename=i18n.filename
47. #网页小图标
48. spring.mvc.favicon.enabled=true

51. #日志
52. #在控制台输出的日志的格式
53. logging.pattern.console=%d{yyyy-MM-dd HH:mm:ss} [%thread] %highlight(%-5level) %logger{50} - %msg%n
54. # 指定文件中日志输出的格式
55. logging.pattern.file=%d{yyyy-MM-dd HH:mm:ss} === [%thread] === %-5level === %logger{50} ==== %msg%n
56. # 指定log输出文件
57. logging.file=my.log
58. #指定log输出文件路径，和logging.file不能一起用
59. #logging.path=xx
60. #设置指定logger的输出等级
61. logging.level.unity.controller.UnityExceptionHandler=info
62. #设置日志的配置文件
63. #logging.config=classpath:logback-test.xml
64. #设置日志的最大保存时间（单位：天）
65. logging.file.max-history=2
66. #######################SpringBoot配置结束##############################


70. ##################项目自定义配置#########################################
71. #查询设置
72. #搜索结果页的显示结果数量
73. myindex.artitleSum=9
75. #Luence索引存放地址
76. lucene.adress=E:/Java Project/Mylucene
77. #lucene.adress=/home/qiu/Myblog
79. #实验图片保存地址
80. image.address=E:/Java Project/unity\_image/
81. #实验保存地址
82. work.address=E:/Java Project/unity\_image/
84. #使用案例图片保存地址
85. index.image=E:/Java Project/unity\_image/
86. #######################项目自定义配置结束################################

3、message\_xx.properties文件

以message为开头的资源文件为国际化的配置文件，在本项目里提供了中文和英文两种语言环境。

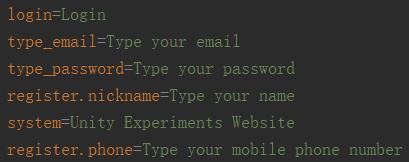
messages.properties为默认的配置，即中文，和messages\_zh\_CN.properties相同

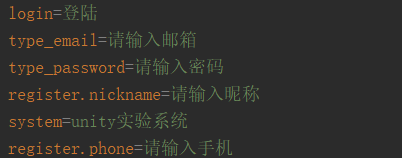
messages\_zh\_CN.properties为中文环境配置

messages\_en\_US.properties为英文环境配置

在本项目中，html页面通过#{}标签即可自动根据语言环境，选择对应的文本。

下面为部分中英文对比





### 5.1.3 业务逻辑控制流程

业务逻辑控制流程如图5-2所示

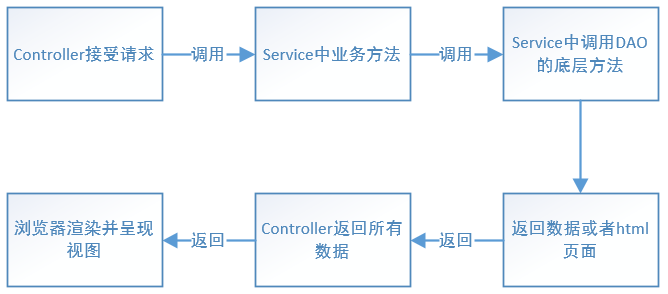


图5-2 逻辑控制流程图

## 5.2 前台功能模块实现

### 5.2.1 注册模块

1、功能描述：用户提供个人信息，网站记录该信息并存入数据库，使其成为注册用户。

2、功能实现：主要页面为register.html，页面提供多个input输入框，收集用户的邮箱、昵称、密码等主要信息，其中邮箱和密码最为重要，是用户登陆的凭证，用户类型主要分为创作者和普通用户两种。当收集到用户的个人信息时，还要进行前端和后端的信息校检，主要有输入信息是否为空、用户是否已注册、两次密码输入是否相同、验证码是否正确，通过校检之后才将密码通过MD加密后连同其他的信息，直接通过调用底层数据库方法，将数据录入，完成注册过程。

代码位于controller包下的UserController.java

部分核心代码如下

@RequestMapping(value = "/register",method = RequestMethod.*GET*)  
public String toregister(){  
 return "register";  
}

*//点击首页导航条的注册链接，即/register，就会进入注册页面*

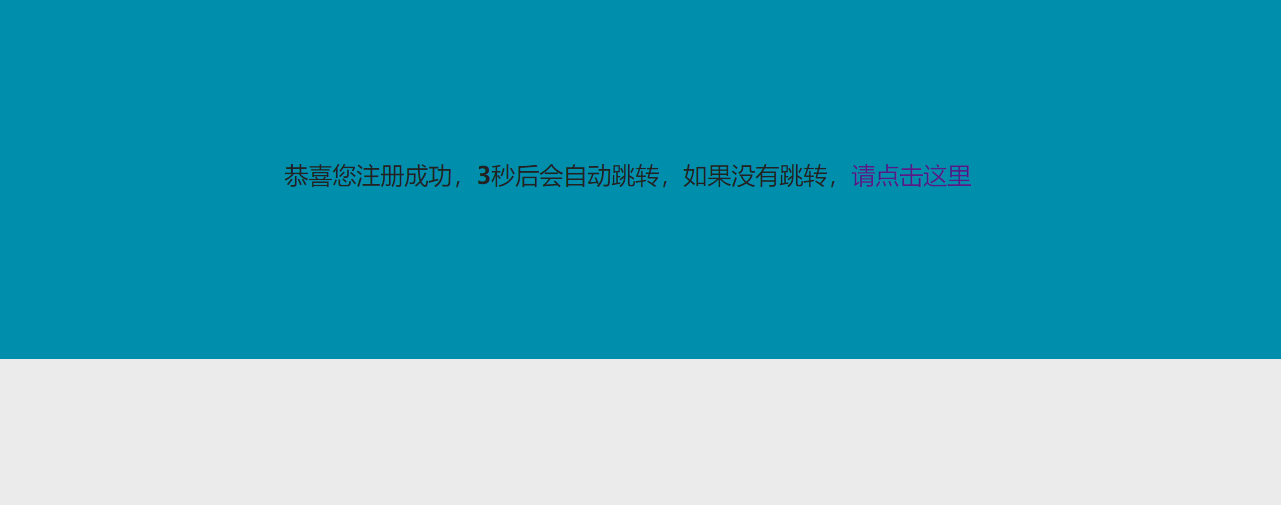
*//然后在注册页面点击注册按钮，即/toregister链接，就会进入下面的注册方法*  
  
@RequestMapping(value = "/toregister",method = RequestMethod.*POST*)  
public String register(User user, Model model, HttpServletRequest request){  
 String email=user.getEmail();

*//这里Md5Hash类是shrio框架的方法，用来将密码通过MD5加密，保存在数据库中*  
 String password=new Md5Hash(user.getPassword(),"unity").toString();  
 if(userService.getUserByEmail(email)!=null){  
 Locale locale=LocaleContextHolder.*getLocale*();  
 String errorInfo=messageSource.getMessage("register.isexist",null,locale);  
 model.addAttribute("errorInfo",errorInfo);

*//这里的操作说明要是输入的邮箱已经存在数据库中，说明已经注册过了，返回提示用户已存在的信息，使用locale的国际化来根据语言环境显示对应语言的提示信息* return "register";  
 }else{  
 Date date=new Date();  
 DateFormat dateFormat=new SimpleDateFormat("yyyy/MM/dd");  
 String time=dateFormat.format(date);  
 user.setRegistrationtime(time);  
 user.setPassword(password);  
 userService.addUser(user);  
 request.getSession().setAttribute("currentUser",user);

*//这里说明所有信息都没有错，将注册日期字段格式化一下，然后填加到数据库中，同时在session中保存User类型的字段，即保存登陆状态，然后跳转到注册成功提示页面*  
 }  
 return "registersuccess"; *//跳转到注册成功页面*}

**注册成功页面：**



*//这里主要都是通过js实现，页面是registersuccess.html，这个页面会显示一个4秒钟的倒计时，用于提醒注册成功。通过window.setTimeout方法设置一个1秒钟的倒计时，每过完一秒又会重新设置1秒中的倒计时并显示倒计时还剩余的时间，直到倒计时4秒全部完成，然后通过window.location.href='/'语句跳转到首页。 核心js如下*

<script type="text/javascript">  
 $(**function** () {  
 **var** time = 4;  
 window.setTimeout(returnUrlByTime, 1000);  
  
 **function** returnUrlByTime() {  
 window.setTimeout(returnUrlByTime, 1000);  
 console.log(time);  
 time = time - 1;  
 **if**(time<=0){  
 time = 0;  
 window.setTimeout("window.location.href='/';window.event.returnValue=false;", 0);  
 }  
 document.getElementById("layer").innerHTML = time;  
 }  
 });

</script>

3、效果图：

效果如图5-3所示



图5-3 注册页面

### 5.2.2 登陆模块

1、功能描述：用户提供邮箱和密码，后台响应完成后用户登陆网站。

2、功能实现：主要页面为login.html，页面主要提供邮箱和密码两个，当用户点击登陆按钮时,首先校检邮箱和密码是否输入，然后进入后端，之后的操作交给Shrio框架，先是判断邮箱和密码是否输入正确，获取到特定用户的信息，然后进行身份认证，判断是创作者还是普通用户，然后进行身份表示，最后保存登陆状态，将用户信息存入Session中。代码依然在UserController.java中。部分核心代码如下：

*//通过点击首页导航栏的登陆链接，即/login，就会跳转到登陆页面*

@RequestMapping(value = "/login",method = RequestMethod.*GET*)  
public String tologin(){  
 return "login";  
}

*//在登陆页面中点击登陆按钮，即/tologin链接，就会进入登陆方法中执行登陆操作*

@RequestMapping(value = "/tologin", method = RequestMethod.*POST*)  
public String login(User user, Model model){

*//获取登录实体* Subject subject = SecurityUtils.*getSubject*();  
  
 *//将输入的密码用MD5加密* String md5string = new Md5Hash(user.getPassword(),"unity").toString();  
  
 *//创建获取用户密码登录token* UsernamePasswordToken token = new UsernamePasswordToken(user.getEmail(), md5string);  
 try {  
 *//根据token登录 会调用MyRealm中的doGetAuthenticationInfo方法进行身份认证，详情看下面的代码解析* subject.login(token);  
 return "redirect:/";  
 } catch (AuthenticationException e) {  
 *//有异常说明登陆出现问题* model.addAttribute("user", user);  
 model.addAttribute("errorInfo", "用户名或密码错误");  
 return "login";  
 }  
}

**Realm包下的MyRealm.java文件：**

*//这个是身份认证方法，在上面的代码中，当执行到subject.login(token);这一句时就会调用到这个方法，参数authenticationToken即传入的token。*

@Override

protected AuthenticationInfo doGetAuthenticationInfo(AuthenticationToken authenticationToken) throws AuthenticationException{

UsernamePasswordToken token=(UsernamePasswordToken) authenticationToken;

*//获取传入的用户邮箱及密码*

String email = (String) authenticationToken.getPrincipal();

String password = new String(token.getPassword());

*//从数据库查询用户信息*

user = userService.getUserByEmail(email);

*//如果邮箱存在，且密码正确，就允许登陆，并在session中保存登陆状态*

if (user != null&&user.getPassword().equals(password)) {

SecurityUtils.getSubject().getSession().setAttribute("currentUser", user);*//把当前用户存到session中*

AuthenticationInfo authcInfo = new SimpleAuthenticationInfo(

email, password, "MyRealm");

return authcInfo;

} else {

*//假如邮箱不存在（未注册用户）或者密码错误就返回空，会使得controller处报异常，说明没有找到账号信息*

return null;

}

}

*//这个方法是权限认证方法，即登陆成功后，每种用户（普通用户，创作者，管理员）都会又自己的权限，权限设置在shrio的配置文件中（config包下的ShrioConfigration.java），下面会详细说明这个文件*

@Override  
protected AuthorizationInfo doGetAuthorizationInfo(PrincipalCollection principalCollection) {  
  
 String email=(String)SecurityUtils.*getSubject*().getPrincipal();  
 SimpleAuthorizationInfo simpleAuthorizationInfo=new SimpleAuthorizationInfo();  
 if(user==null){  
 user=userService.getUserByEmail(email);  
 }  
 if(user.getUsertype()!=0){ *//如果是创作者或者是管理员就为其添加admin权限*  
 Set<String> set=new HashSet<>();  
 set.add("admin");  
 simpleAuthorizationInfo.setRoles(set);  
 }  
 return simpleAuthorizationInfo;  
}

**config包下的ShrioConfigration.java：**

@Bean  
public ShiroFilterFactoryBean shirFilter(SecurityManager securityManager){  
  
 ShiroFilterFactoryBean shiroFilterFactoryBean = new ShiroFilterFactoryBean();  
  
  *// 必须设置 SecurityManager*  
 shiroFilterFactoryBean.setSecurityManager(securityManager);  
 *// 如果不设置默认会自动寻找Web工程根目录下的"/login.jsp"页面* shiroFilterFactoryBean.setLoginUrl("/login");  
 *// 登录成功后要跳转的链接* shiroFilterFactoryBean.setSuccessUrl("/");  
 *//拦截器.*  
 Map<String,String> filterChainDefinitionMap = new LinkedHashMap<String,String>();  
  *//自定义拦截器(失败拦截）*  
 Map<String, Filter> customisedFilter = new HashMap<>();  
  
*//下面的映射关系配置说明，第一个参数即路径链接，第二个则是权限。*

*//anon权限的意思是无权限，任何用户都可以访问*

*//authc说明只要是登陆了的用户就可以方法*

*//roles[xxx]说明不仅要登陆，还必须要拥有xxx权限的用户才可以访问*

*//其他更多的相关信息请参考Shrio官网*  
  *//配置映射关系（这里顺序很重要）* filterChainDefinitionMap.put("/login", "anon");  
 filterChainDefinitionMap.put("/static/\*\*", "anon");  
  *//后台的管理需要有admin的角色,这里就是前面的权限认证方法的作用，如果没有admin权限而强行进入/backstage链接，那么就会报错*  
 filterChainDefinitionMap.put("/backstage", "roles[admin]");  
 filterChainDefinitionMap.put("/admin/\*\*", "authc"); //roles[admin]  
  *//登陆的用户才能评论*  
 filterChainDefinitionMap.put("/assess", "authc");  
 filterChainDefinitionMap.put("/\*\*", "anon");  
 shiroFilterFactoryBean.setFilters(customisedFilter);  
 shiroFilterFactoryBean.setFilterChainDefinitionMap(filterChainDefinitionMap);  
 *logger*.info("Shrio拦截器链配置成功！！");  
 return shiroFilterFactoryBean;  
}

3、效果图

效果如图5-4所示



图 5-4 登陆页面

### 5.2.3 导航条模块

1、功能描述：导航条起到为用户导航的作用，其至始至终都会显示在前台页面的最上方，提供返回主页，登陆注册，显示登陆状态、创作者进入后台以及搜索框功能。

2、功能实现：页面为head.html，主要是通过前端html+css+js以及Thymeleaf模板引擎的方式实现，其中实现页面跳转的功能如返回主页，登陆注册等是通过<a></a>标签实现，而显示登陆状态是通过thymeleaf的${}标签实现，根据session中是否保存有用户信息而显示对应的信息。

导航条都是通过html和js以及thymleaf框架实现的，导航条html文件为head.html。

**head.html文件：**

1. <!DOCTYPE html**>**
2. **<html**  xmlns:th="http://www.thymeleaf.org" xmlns="http://www.w3.org/1999/html"**>**
3. **<head>**
4. **<meta** charset="UTF-8"**>**
5. **<title>**head**</title>**    <!--通用头部-->
6. **</head>**
7. **<body>**
9. **<nav** class="navbar navbar-default navbar-fixed-top" style="background:#171717;"  th:fragment="headbar"**>**
11. **<div** class="container-fluid"**>**
12. <!-- Brand and toggle get grouped for better mobile display -->
13. **<div** class="navbar-header mynav"**>**
14. **<a** class="navbar-brand" href="/"**>**
15. **<span** class="glyphicon glyphicon-home" aria-hidden="true"**></span>** 第一虚拟课堂
16. **</a>**
17. **</div>**
19. <!-- Collect the nav links, forms, and other content for toggling -->
20. **<div** class="collapse navbar-collapse" id="bs-example-navbar-collapse-1"**>**
21. **<ul** class="nav navbar-nav"**>**
22. **<li** th:if="${(session.currentUser!=null)&&(session.currentUser.usertype!=0)}"**><a** href="/backstage" th:text="#{head.backstage}"**>**进入后台**</a></li>**
23. **</ul>**
24. **<form** class="navbar-form navbar-right" id="queryfrom" onkeydown="cdk()"**>**
25. **<div** class="form-group"**>**
26. **<input** type="text" id="queryword" class="form-control searchinput" th:placeholder="#{head.search}"**>**
27. **</div>**
28. **<a** href ="javascript:search()" class="searchbtn" id="searchbtn"**></a>**
29. **</form>**
30. **<ul** class="nav navbar-nav navbar-right" th:if="${session.currentUser!=null}"**>**
31. **<li><a** href="/logout" th:text="#{head.logout}"**>**退出登陆**</a></li>**
32. **</ul>**
33. **<ul** class="nav navbar-nav navbar-right"**>**
34. **<li>**
35. **<a** th:href="${session.currentUser!=null}?'#':'/login'"**>**
36. **<p** th:text="${session.currentUser!=null}?${session.currentUser.username}:#{head.login}"**></p>**
37. **</a>**
38. **</li>**
40. **<li>**
41. **<a** th:if="${session.currentUser==null}" th:href="@{/register}"**>**
42. **<p** th:text="#{register}"**></p>**
43. **</a>**
44. **</li>**
45. **</ul>**
46. **<ul** class="nav navbar-nav navbar-right"**>**
47. **<li** class="dropdown"**>**
48. **<a** href="#" class="dropdown-toggle" data-toggle="dropdown" role="button" aria-haspopup="true" aria-expanded="false" th:text="#{head.change}"**>**语言切换 **<span** class="caret"**></span></a>**
49. **<ul** class="dropdown-menu"**>**
50. **<li><a** th:href="@{/changelang(language='zh')}" th:text="#{head.lang1}"**>**中文**</a></li>**
51. **<li** role="separator" class="divider"**></li>**
52. **<li><a** th:href="@{/changelang(language='en')}" th:text="#{head.lang2}"**>**英文**</a></li>**
53. **</ul>**
54. **</li>**
55. **</ul>**
56. **</div>**
57. **</div>**
58. **</nav>**
60. **</body>**
61. **</html>**

*首先要使用thymleaf框架的标签，就必须在第2行处写上xmlns:th="http://www.thymeleaf.org" 。*

*第9行中出现了 th:fragment="headbar"标签，意思是给整个nav标签命名为headbar，然后再其他的html文件中就可以引用这个nav了，也就是说实现了导航栏在其他文件中的复用（前台的所有页面都要又这个导航栏）。*

*第22行为进入后台的链接，但是因为普通用户是没有后台这一权限的，而且必须要登陆了的用户才有可能进入后台，所以这里使用th:if="${(session.currentUser!=null)&&(session.currentUser.usertype!=0)}"判断是否已经登陆，而且并不是普通用户，那么就显示这个进入后台链接*

*第24行开始的一个from表单提供的是搜索框的功能，其中onkeydown="cdk()"是search.js文件中的方法，在这里的意思是当按下回车键是就会进行关键词搜索，这个功能具体在下文的搜索框模块。*

*第36说明当未登录时，会显示登陆按钮，当已经登陆时，就显示用户的用户名*

*第30和31行说明当登陆之后，就是显示退出登陆的链接*

如根据登陆状态确定是否显示注册页面代码如下：

<li>  
 <a th:if="${session.currentUser==null}" th:href="@{/register}">  
 <p th:text="#{register}"></p>  
 </a>  
</li>

3、效果图

未登陆效果如图5-5所示，登陆后效果如图5-6所示。



图5-5 未登陆



图5-6 已登陆

### 5.2.4 国际化模块

1、功能描述：国际化即语言切换，本网站在导航条处提供了中英文的切换功能。

2、功能实现：国际化功能的实现主要通过Thymeleaf模板引擎，在Thymeleaf中，#{}标签能够根据当前的语言环境，从对应的资源文件中选择文本。资源文件为.properties格式，里面是形如“register=注册”的键值对。

**controller包下的LangController.java文件：**

*//在之前的导航条模块中提供了切换中英文的操作，链接即 /changelang*

*//代码中用到了一个类叫做SessionLocaleResolver，顾名思义就是保存在session中的国际化解析器，这个类会在unity包下的Application.java启动类中被注入。通过在session中保存国际化环境就能够实现实时的切换语言环境。*

@RequestMapping("/changelang")  
public String changlang(HttpServletRequest request, String language){  
 if("zh".equals(language)) {  
 request.getSession().setAttribute(  
 SessionLocaleResolver.*LOCALE\_SESSION\_ATTRIBUTE\_NAME*,  
 new Locale("zh","CN"));  
 }else if("en".equals(language)){  
 request.getSession().setAttribute(  
 SessionLocaleResolver.*LOCALE\_SESSION\_ATTRIBUTE\_NAME*,  
 new Locale("en","US"));  
 }  
 return "redirect:/";  
}

**Application.java启动类文件：**

*//使用@Bean主键注入一个SessionLocaleResolver*

@Bean  
public LocaleResolver localeResolver() {  
 SessionLocaleResolver slr =new SessionLocaleResolver();  
 //设置默认区域,  
 slr.setDefaultLocale(Locale.*CHINA*);  
 return slr;  
}

**网页是如何调用的：**

**主要有两种方式：**

1. **在html页面中，通过#{xxx}标签即可，“xxx”就是message.properties中的键值**
2. **在html页面中，通过 [[#{xxx}]] 即可，意义同上，区别就是外边有双方括号**

3、效果图：

切换为英文的效果如图5-7所示

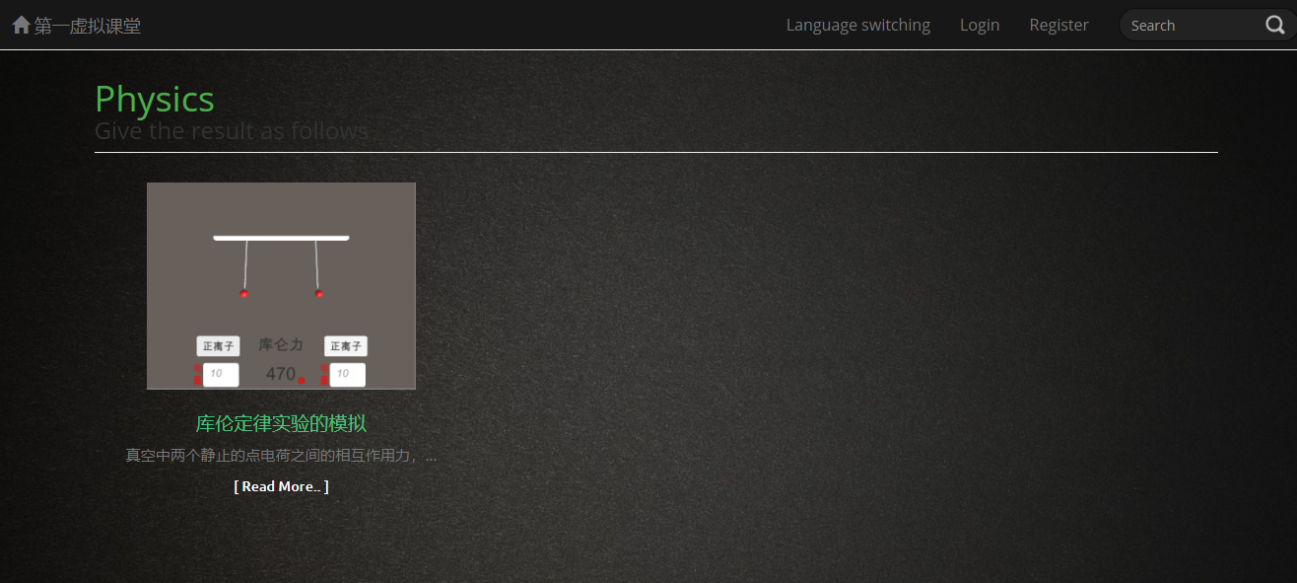


图5-7 国际化

### 5.2.5 搜索框模块

1、功能描述：用户在搜索框中键入实验的关键词，即可搜索到相关的实验。

2、功能实现：该功能在导航条中实现，底层使用了Luence全文搜索引擎，在创建实验的时候，对实验的标题、关键词和实验描述进行分词，通过倒排索引的方式保存到Luence的索引库中，当使用搜索框的时候，针对输入的词语或短语，使用倒排索引反向查到实验的主键，完成查询过程。

**搜索框主要用到search.js:**

function search() {  
 var queryword=$("#queryword").val();  
 //debugger;  
 if(queryword.length==0){  
 window.location.href='http://localhost:8088'; window.event.returnValue=false;

//如果没有输入关键词，就直接跳转到主页  
 }else{  
 $("#queryfrom").attr("action","/g/search?queryword="+queryword);  
 $("#queryfrom").attr("method","post").submit();

*//点击搜索按钮，就会将输入的关键词加到/g/search链接后面，通过post方式提交到后台* }  
}

*//cdk方法作用为，当按下回车键时，进入搜索过程。。。*  
function cdk(){  
 if(event.keyCode ==13){  
 search();  
 }  
}

**controller包下的WorkController.java:**

@RequestMapping(value = "/g/search")  
public String showWorksBySearch(@RequestParam("queryword") String queryword,  
 @RequestParam(value = "page",defaultValue = "1")int page,  
 Model model)throws Exception{  
  
 //*todo 考虑到数量不多，暂时采用最笨的方法分页* List<Work> list=workIndex.searchWork(queryword);  
*//因为有分页处理，所以要判断当前是第几页，是否还有下一页，总的页数* int count=list.size();  
 long allpages=count%pagesum==0?(count/pagesum):(count/pagesum+1);  
  
 int fromindex=(page - 1) \* pagesum;  
 int toIndex = count>= page \* pagesum ? page\* pagesum : count;  
  
 List<Work> worklist=list.subList(fromindex,toIndex);  
  
 List<Workvo> workresults=new ArrayList<>();  
 for(Work work:worklist){  
 Workvo workvo= workvoServiceImpl.pojoTovo(work, WorkvoServiceImpl.*WORKSRESULT\_MODE*);  
 workresults.add(workvo);  
 }  
 Long startpage= workvoServiceImpl.pageprocess(page,pagesum);  
  
 model.addAttribute("workresults",workresults);  
 model.addAttribute("allpages",allpages);  
 model.addAttribute("currentpage",page);  
 model.addAttribute("startpage",startpage);  
 model.addAttribute("queryword",queryword);  
 return "worksresult";  
}

**config包下的WorkIndex.java：**

//搜索功能主要通过Lucene框架，通过lucene库查询

public List<Work> searchWork(String q) throws Exception{  
 //实例化目录对象  
 directory = FSDirectory.*open*(Paths.*get*(indexDir));  
 //获取读索引对象  
 IndexReader indexReader = DirectoryReader.*open*(directory);  
 //获取索引查询对象  
 IndexSearcher indexSearcher = new IndexSearcher(indexReader);  
 //组合查询对象  
 BooleanQuery.Builder builder = new BooleanQuery.Builder();  
 //中文分析器  
 SmartChineseAnalyzer chineseAnalyzer = new SmartChineseAnalyzer();  
  
 //标题查询分析器  
 QueryParser titleParser = new QueryParser("workname",chineseAnalyzer);  
 //标题查询器  
 Query titleQuery = titleParser.parse(q);  
  
 //关键词查询分析器  
 QueryParser keywordParser = new QueryParser("keyword",chineseAnalyzer);  
 //关键词查询器  
 Query keywordQuery = keywordParser.parse(q);  
  
 //内容查询分析器  
 QueryParser contentParser = new QueryParser("content",chineseAnalyzer);  
 //内容查询器  
 Query contentQuery = contentParser.parse(q);  
  
 //添加标题查询器 逻辑关系为或者  
 builder.add(titleQuery, BooleanClause.Occur.*SHOULD*);  
 //添加关键词查询器 逻辑关系为或者  
 builder.add(keywordQuery, BooleanClause.Occur.*SHOULD*);  
 //添加内容查询器 逻辑关系为或者  
 builder.add(contentQuery, BooleanClause.Occur.*SHOULD*);  
  
 //查询前100条记录放在his中  
 TopDocs hits = indexSearcher.search(builder.build(),100);  
 //将title得分高的放在前面  
 //QueryScorer queryScorer = new QueryScorer(titleQuery);  
 //得分高的片段  
 //Fragmenter fragmenter = new SimpleSpanFragmenter(queryScorer);  
 //格式化得分高片段  
 //SimpleHTMLFormatter formatter = new SimpleHTMLFormatter("", "");  
 //高亮显示  
 //Highlighter highlighter = new Highlighter(formatter,queryScorer);  
 //将得分高的片段设置进去  
 //highlighter.setTextFragmenter(fragmenter);  
 //用来封装查询到的博客  
 List<Work> workList = new LinkedList<>();  
 //遍历查询结果  
 for(ScoreDoc score : hits.scoreDocs) {  
 Document doc = indexSearcher.doc(score.doc);  
 Work work = workService.getWork(Integer.*parseInt*(doc.get("id")));  
 workList.add(work);  
 }  
 return workList;  
}

**搜索框的前端实现：**

head.html:

1. **<form** class="navbar-form navbar-right" id="queryfrom" onkeydown="cdk()"**>**
2. **<div** class="form-group"**>**
3. **<input** type="text" id="queryword" class="form-control searchinput" th:placeholder="#{head.search}"**>**
4. **</div>**
5. **<a** href ="javascript:search()" class="searchbtn" id="searchbtn"**></a>**
6. **</form>**

**Css中关于搜索框属性：**

head.css:

1. .searchinput{
2. **border**:1px solid #0c0c0c;
3. border-radius: 50px;
4. **background-color**: #222;
5. **display**:inline;
6. }
8. .searchbtn{
9. **width**:200px;
10. **height**:200px;
11. top:-75px;
12. right: -60px;
13. **background**:url(../images/sousuo.png) no-repeat;
14. transform: scale(0.1);
15. **position**: absolute;
16. }

border属性设置搜索框的边框粗细，形状以及颜色，border-radius给搜索框设置一个圆角。

. searchbtn 这个class设置了搜索案件的图片外观、形状以及位置，transform属性为它的缩放比例。

3、效果图



### 5.2.6 首页模块

1、功能描述：首页主要提供三大类的实验链接，以及展示网站公告和使用案例。

2、功能实现：主要页面为index.html，通过html+css实现，实验链接通过<a>标签实现，根据实验类型在后端中查询，而网站和使用案例则直接通过简单的数据库查询后显示。

**关键Index.html:**

1. **<div** th:replace="head :: headbar"**>**
2. **</div>**
4. **<div** class="banner"**>**
5. **<div** class="header"**>**
6. **<div** class="clearfix"**></div>**
7. **</div>**
9. **<div** class="slider"**>**
10. **<div** class="caption"**>**
11. **<div** class="container"**>**
12. **<div** class="callbacks\_container"**>**
13. **<ul** class="rslides" id="slider"**>**
14. **<li><h3>**第一虚拟课堂主要采用unity3d引擎，提供交互式的实验体验**</h3></li>**
15. **<li><h3>**第一虚拟课堂囊括了物理、化学以及生物类实验，基本满足所有需求**</h3></li>**
16. **<li><h3>**虚拟实验的目的就是让更多的同学在教室中享受到“身临其境”的实验感觉**</h3></li>**
17. **</ul>**
18. **<div** class="clearfix"**></div>**
19. **</div>**
20. **</div>**
21. **</div>**
22. **</div>**
24. **<div** class="banner-grids"**>**
25. **<div** class="container"**>**
26. **<div** class="col-md-4 banner-grid"**>**
27. <!--<h3 style="background:#40a6e7;" th:text="#{index.Chemistry}">化学</h3>-->
28. **<div** class="banner-grid-sec" style="height: 253px;"**>**
29. **<a** href="/g/type/1"**><img** th:src="@{/static/images/cpjs\_huaxue.jpg}" style="width:100%;height:100%;"**/></a>**
30. **<h3** style="position: absolute; top: 36px; left: 36px;" th:text="#{index.Chemistry}"**></h3>**
31. **</div>**
32. **</div>**
33. **<div** class="col-md-4 banner-grid"**>**
34. <!--<h3 style="background:#19191e;" th:text="#{index.Physics}">物理</h3>-->
35. **<div** class="banner-grid-sec"**>**
36. **<a** href="/g/type/2"**><img** th:src="@{/static/images/cpjs\_wuli.jpeg}" style="width:100%;height:100%;"**/></a>**
37. **<h3** style="position: absolute; top: 36px; left: 36px;" th:text="#{index.Physics}"**></h3>**
38. **</div>**
39. **</div>**
40. **<div** class="col-md-4 banner-grid"**>**
41. <!--<h3 style="background:#388d00;" th:text="#{index.Biology}">生物</h3>-->
42. **<div** class="banner-grid-sec"**>**
43. **<a** href="/g/type/3"**><img** th:src="@{/static/images/cpjs\_shengwu.jpg}" style="width:100%;height:100%;"**/></a>**
44. **<h3** style="position: absolute; top: 36px; left: 36px;" th:text="#{index.Biology}"**></h3>**
45. **</div>**
46. **</div>**
47. **<div** class="clearfix"**></div>**
48. **</div>**
49. **</div>**
50. **</div>**
52. **<div** class="news"**>**
53. **<div** class="container"**>**
54. **<h3** th:text="#{index.news}"**>**Top News**</h3>**
55. **<div** class="event-grids"**>**
56. **<div** class="col-md-4 event-grid" th:each="news:${newsresult}"**>**
57. **<div** class="date"**>**
58. **<h4** th:text="${#strings.substring(news.ctime,8,10)}"**>**26**</h4>**
59. **<span** th:text="${#strings.substring(news.ctime,0,7)}"**>**08/2012**</span>**
60. **</div>**
61. **<div** class="event-info"**>**
62. **<h5** th:text="${news.newsname}"**>**公告**</h5>**
63. **<p** th:text="${news.newscontent}"**>**公告**</p>**
64. **</div>**
65. **<div** class="clearfix"**></div>**
66. **</div>**
67. **<div** class="clearfix"**></div>**
68. **</div>**
69. **</div>**
70. **</div>**

73. **<div** class="welcome"**>**
74. **<div** class="container"**>**
75. **<h2** th:text="#{index.anli}"**>**使用案例**</h2>**
76. **<div** class="welcm\_sec"**>**
77. **<div** class="col-md-12 campus" align="center"**>**
78. **<div** class="campus\_head"**>**
79. **<p>**这里展示的使用的案例**</p>**
80. **</div>**
81. **<div** class="col-md-3 wel\_grid caseimage" th:each="cases:${caseresult}"**>**
82. **<img** th:src="@{/showimage/\_\_${cases.image}\_\_}" class="img-responsive" alt=""**/>**
83. **<h5><a** href="#" th:text="${cases.casename}"**> </a></h5>**
84. **<p** th:text="${cases.content}"**> </p>**
85. **</div>**
86. **<div** class="clearfix"**></div>**
87. **</div>**
88. **<div** class="clearfix"**></div>**
89. **</div>**
90. **</div>**
91. **</div>**

第1、2行即引入导航栏功能

第9-22行为首页滚动的提示字符，是硬编码在首页html文件之中的。

第24-49行为三大分类的展示，主要就是分类图片和链接，其中文字显示在图片上的实现效果为：

position: absolute; top: 36px; left: 36px;

通过设置文字位置为绝对位置，改变文字的位置即可实现。

第20-70行为公告位置，第73-90行为使用案例展示，他们的内容都是通过“${}”标签取得，使用案例的图片通过“@{}”标签取得，如th:src="@{/showimage/\_\_${cases.image}\_\_}"

其中showimage为此前在config包里的MySpringMVCConfig.java文件中配置的本地路径。

**Controller包里的IndexController.java:**

1. /\*\*
2. \* 展示首页
3. \* @param model
4. \* @return
5. \*/
6. @RequestMapping("/")
7. **public** String showindex(Model model){
8. List<News> newslist=(ArrayList)newsMapper.selectAll();
9. List<Cases> caselist=(ArrayList)casesMapper.selectAll();
10. **if**(newslist.size()<3){
11. model.addAttribute("newsresult",newslist);
12. }**else**{
13. List<News> newsresult= newslist.subList(0,3);
14. model.addAttribute("newsresult",newsresult);
15. }
16. **if**(caselist.size()<4){
17. model.addAttribute("caseresult",caselist);
18. }**else**{
19. List<Cases> caseresult= caselist.subList(0,4);
20. model.addAttribute("caseresult",caseresult);
21. }
22. **return** "index";
23. }

首页的后台处理非常简单，除了公告和使用案例展示都是硬编码。这里的处理是根据时间顺序取处前3个公告和前4个使用案例，可以根据情况修改。

3、效果图

首页效果如图5-8和图5-9所示：



图5-8 首页图（1）



图5-9 首页图（2）

### 5.2.7 结果页模块

1、功能描述：根据分类查询或者关键词查询，展示相关实验

2、功能实现：该功能在workresult.html页面中实现。分类查询通过实验的实验类型字段，而关键词查询则是通过Luence全文检索引擎搜索。最后将得到的搜索结果通过Thymleaf模板引擎的th:each标签遍历所有的结果，然后使用${}标签展示结果，如果结果过多，还要进行分页处理。

**worksresult.html：**

1. <!DOCTYPE HTML**>**
2. **<html** xmlns:th="http://www.thymeleaf.org"**>**
3. **<head>**
4. **<title** th:text**=**"#{worksresult.title}"**>**结果**</title>**
6. **<link** rel="stylesheet" th:href="@{/static/bootstrap-3.3.7/css/bootstrap.css}" type="text/css"**/>**
7. **<link** th:href="@{/static/css/worksresult.css}" rel='stylesheet' type='text/css' **/>**
8. **<link** th:href="@{/static/css/head.css}" rel='stylesheet' type='text/css' **/>**
10. **<meta** name="viewport" content="width=device-width, initial-scale=1"**>**
11. **<meta** http-equiv="Content-Type" content="text/html; charset=utf-8" **/>**

14. **<script** th:src="@{/static/layui/jquery-3.3.1.min.js}"**></script>**
15. **<script** th:src="@{/static/bootstrap-3.3.7/js/bootstrap.js}"**></script>**
16. **<script** th:src="@{/static/js/search.js}"**></script>**

19. **</head>**
21. **<body>**
23. **<input** type="text" th:unless="${#strings.isEmpty(queryword)}" th:hidden="true" th:value="${queryword}" id="isquery"**/>**
25. **<div** th:replace="head :: headbar"**>**
26. **</div>**

29. **<div** class="about-sec"**>**
30. **<div** class="container"**>**
32. **<div** class="page-header" style="margin-top:5em" th:unless="${#strings.isEmpty(queryword)}"**>**
33. **<h1** style="color: #4cae4c;"  th:text="${queryword}"**></h1><h3>**[[#{worksresult.result}]]**</h3>**
34. **</div>**
35. **<div** th:if="${#lists.size(workresults)==0}" class="noinfo"**>**[[#{worksresult.noinfo}]]**</div>**
36. **<div** class="our\_work"**>**
37. **<div** class="col-md-4 camp-grid" th:each="workvo:${workresults}" align="center"**>**
38. **<a** th:href="${workvo.forfree==1}?'/g/id/'+${workvo.workid}:''"**>**
39. **<div** class="workimage"**>**
40. **<img** th:unless="${workvo.indeximage==null}" th:src="@{/showimage/\_\_${workvo.indeximage}\_\_}" class="img-responsive"**/>**
41. **<img** th:if="${workvo.indeximage==null}" th:src="@{/static/images/noimage.png}" class="img-responsive" alt=""**/>**
42. **</div>**
43. **</a>**
44. **<h4** th:text="${workvo.workname}" th:if="${workvo.forfree==1}"**>**$$$**</h4>**
45. **<del** th:text="${workvo.workname}" th:if="${workvo.forfree!=1}"**>**$$$**</del>**
46. **<p** th:text="${workvo.content}"**>**$$$**</p>**
47. **<a** th:href="${workvo.forfree==1}?'/g/id/'+${workvo.workid}:''" class="more"**>**[ [[#{worksresult.readmore}]] ]**</a>**
48. **</div>**
49. **</div>**

52. **<div**  th:if="${#lists.size(workresults)}==9" class="col-md-12" align="center"**>**
53. **<ul** class="pagination"**>**
54. **<li** th:class="${currentpage==1}?'disabled':'btn'"**>**
55. **<a** th:attr="href='?page=1'" aria-label="Previous" th:class="${currentpage==1}?'btn disabled':'btn'"**>**
56. **<span** aria-hidden="true"**>**«**</span>**
57. **</a>**
58. **</li>**
59. **<li** th:class="${currentpage==1}?'disabled':'btn'"**>**
60. **<a** th:attr="href='?page='+${currentpage-1}" th:class="${currentpage==1}?'btn disabled':'btn'"**>**
61. **<span** aria-hidden="true"**><</span>**
62. **</a>**
63. **</li>**
65. **<li** th:if="${startpage<=allpages}"**><a** th:attr="href='?page='+${startpage}" th:text="${startpage}" class="btn"**></a></li>**
66. **<li** th:if="${startpage+1<=allpages}"**><a** th:attr="href='?page='+${startpage+1}" th:text="${startpage+1}" class="btn"**></a></li>**
67. **<li** th:if="${startpage+2<=allpages}"**><a** th:attr="href='?page='+${startpage+2}" th:text="${startpage+2}" class="btn"**></a></li>**
68. **<li** th:if="${startpage+3<=allpages}"**><a** th:attr="href='?page='+${startpage+3}" th:text="${startpage+3}" class="btn"**></a></li>**
69. **<li** th:if="${startpage+4<=allpages}"**><a** th:attr="href='?page='+${startpage+4}" th:text="${startpage+4}" class="btn"**></a></li>**
71. **<li** th:class="${currentpage==allpages}?'disabled':''"**>**
72. **<a** th:attr="href='?page='+${currentpage+1}" th:class="${currentpage==allpages}?'btn disabled':'btn'"**>**
73. **<span** aria-hidden="true"**>></span>**
74. **</a>**
75. **</li>**
76. **<li** th:class="${currentpage==allpages}?'disabled':''"**>**
77. **<a** th:attr="href='?page='+${allpages}" aria-label="Next" th:class="${currentpage==allpages}?'btn disabled':'btn'"**>**
78. **<span** aria-hidden="true"**>**»**</span>**
79. **</a>**
80. **</li>**
81. **</ul>**
82. **</div>**
83. **</div>**
84. **</div>**
86. **<script>**
87. //关键字查询时为分页带上参数
88. $(function () {
89. var queryword=$("#isquery").val();
90. if(queryword!=undefined){
91. var length=$(".pagination a").length;
92. var aa=$(".pagination a");
93. for(var i=0;i**<length**;i++){
94. ahref=$(aa[i]).attr("href");
95. $(aa[i]).attr("href",ahref+"&queryword="+queryword);
96. }
97. }
98. })
99. **</script>**
101. **</body>**
102. **</html>**

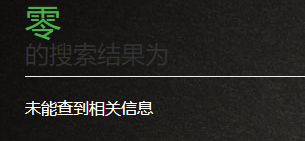
*结果页实现了复用，在关键词搜索，分类搜索中都 使用该页面。*

*第23-24行使用了一个隐藏域（th:hidden="true"），用来保存查询时的关键词，在结果页的32-34行中会显示出来，通过判断关键词是否为空，非空就显示，如下：*



*第25行导入导航条*

*第35-36行通过“th:if="${#lists.size(workresults)==0}"”判断搜索结果是否为空，如果搜索结果为空就显示空提醒，如：*



*第39-49行为所有搜索结果的遍历，通过“th:each="workvo:${workresults}"”，此后通过“${workvo}”取到了所有的结果，显示所有的结果信息。*

*第52-84都是分页处理。分页采用的是bootstrap的分页，首先判断是否需要显示分页条（因为我后台规定一页显示9个结果，假如结果小于9就没有必要显示了），然后就是当前的页，还有判断是否还有下一页，在页数最多的情况下，总共有这么9个按钮（回到第一页，前一页，分页按钮开始页数，分页按钮开始页数+1，分页按钮开始页数+2，分页按钮开始页数+3，分页按钮开始页数+4，后一页，去到最后一页）*

*第86-99行是js代码，这是为分页处理的。从65行的“th:attr="href='?page='+${startpage}"”可以看出默认是当前路径后面加入page参数，即当前页参数，但是如果我们是关键词查询而不是分页查询时，就会丢失关键词，所以这里我们判断如果是关键词查询时，点击页码时补上关键词。这里用到了jquery。*

**controller包下的WorkController.java:**

1. /\*\*
2. \* 通过分类获取实验
3. \* @param worktype
4. \* @param page   当前页码
5. \* @param model
6. \* @return
7. \*/
8. @RequestMapping(value = "/g/type/{worktype}",method = RequestMethod.GET)
9. **public** String showWorksByType(@PathVariable Integer worktype,
10. @RequestParam(value = "page",defaultValue = "1")**int** page,
11. Model model){
13. //下面pagehelper必须先打开，再进行查询操作，才能成功分页
14. PageHelper.startPage(page,pagesum,"id desc");
15. List<Work> works=workServiceImpl.getWorksByType(worktype);
16. PageInfo<Work> data=**new** PageInfo<>(works);
17. List<Work> worklist=data.getList();
19. Long count=data.getTotal();
20. **long** allpages=count%pagesum==0?(count/pagesum):(count/pagesum+1);
22. List<Workvo> workresults=**new** ArrayList<>();
23. **for**(Work work:worklist){
24. Workvo workvo= workvoServiceImpl.pojoTovo(work, WorkvoServiceImpl.WORKSRESULT\_MODE);
25. workresults.add(workvo);
26. }
27. Long startpage= workvoServiceImpl.pageprocess(page, pagebuttonsum);
29. Locale locale= LocaleContextHolder.getLocale();
30. String queryword="";
31. **switch** (worktype){
32. **case** 1:
33. queryword=messageSource.getMessage("index.Chemistry",**null**,locale);
34. **break**;
35. **case** 2:
36. queryword=messageSource.getMessage("index.Physics",**null**,locale);
37. **break**;
38. **case** 3:
39. queryword=messageSource.getMessage("index.Biology",**null**,locale);
40. **break**;
41. }
42. model.addAttribute("queryword",queryword);
43. model.addAttribute("workresults",workresults);
44. model.addAttribute("allpages",allpages);
45. model.addAttribute("currentpage",page);
46. model.addAttribute("startpage",startpage);
47. **return** "worksresult";
49. }
50. /\*\*
51. \* 通过名称和关键字模糊查询实验，可以post也可以get方法
52. \* @param queryword
53. \* @param page
54. \* @param model
55. \* @return
56. \* @throws Exception
57. \*/
58. @RequestMapping(value = "/g/search")
59. **public** String showWorksBySearch(@RequestParam("queryword") String queryword,
60. @RequestParam(value = "page",defaultValue = "1")**int** page,
61. Model model)**throws** Exception{
63. //todo 考虑到数量不多，暂时采用最笨的方法分页
64. List<Work> list=workIndex.searchWork(queryword);
66. **int** count=list.size();
67. **long** allpages=count%pagesum==0?(count/pagesum):(count/pagesum+1);
69. **int** fromindex=(page - 1) \* pagesum;
70. **int** toIndex = count>= page \* pagesum ? page\* pagesum : count;
72. List<Work> worklist=list.subList(fromindex,toIndex);
74. List<Workvo> workresults=**new** ArrayList<>();
75. **for**(Work work:worklist){
76. Workvo workvo= workvoServiceImpl.pojoTovo(work, WorkvoServiceImpl.WORKSRESULT\_MODE);
77. workresults.add(workvo);
78. }
79. Long startpage= workvoServiceImpl.pageprocess(page, pagebuttonsum);
81. model.addAttribute("workresults",workresults);
82. model.addAttribute("allpages",allpages);
83. model.addAttribute("currentpage",page);
84. model.addAttribute("startpage",startpage);
85. model.addAttribute("queryword",queryword);
86. **return** "worksresult";
87. }

*可以看到，不管是分类查询还是关键词查询，都会有*

*model.addAttribute("allpages",allpages);*

*model.addAttribute("currentpage",page);*

*model.addAttribute("startpage",startpage);*

*model.addAttribute("queryword",queryword);*

*这么四个参数，其中分类查询时，关键词queryword就是分类名：物理、化学、生物。其他三个为分页处理服务。Allpages是总页数， page是用户点击的页数，startpage是分页按钮开始页数。*

*效果如下：*



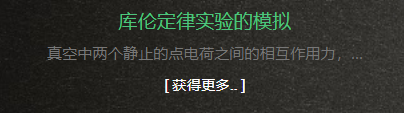
**service包下WorkvoServiceImpl.java：**

1. /\*\*
2. \* 分页时计算起始页
3. \* @param currentpage  当前页数
4. \* @param pagebutton   分页按钮数目
5. \* @return
6. \*/
7. **public** **long** pageprocess(**long** currentpage,**int** pagebutton){
8. **long** startpage;
9. **if**(currentpage%pagebutton==0){
10. startpage=currentpage-pagebutton+1;
11. }**else**{
12. startpage=currentpage%pagebutton==1?currentpage:(currentpage+1-currentpage%pagebutton);
13. }
14. **return** startpage;
15. }

*这个是计算startpage的方法*

1. /\*\*
2. \* 将pojo转为vo
3. \* @param work
4. \* @return
5. \*/
6. **public**  Workvo pojoTovo(Work work,Integer mode) {
8. **if** (mode == WORKSRESULT\_MODE) {     //结果页
9. Workvo workvo = **new** Workvo();
10. String content = work.getContent();
11. **if** (content != **null** && content.length() > 20) {
12. content = content.substring(0, 20)+"...";
13. }
14. workvo.setWorkid(work.getId());
15. workvo.setForfree(work.getForfree());
16. workvo.setContent(content);
17. workvo.setWorkname(work.getWorkname());
18. String images = work.getPictures();
19. **if** (images != **null** && !"".equals(images)) {
20. String[] imagelist = images.split(",");
21. workvo.setIndeximage(work.getWorkmanid() + "/" + imagelist[0]);
22. } **else** {
23. workvo.setIndeximage(**null**);
24. }
25. **return** workvo;
26. } **else** **if** (mode == WORKDETAIL\_MODE) {    //详情页
27. Workvo workvo = **new** Workvo();
28. workvo.setWorkid(work.getId());
29. workvo.setContent(work.getContent());
30. workvo.setWorkname(work.getWorkname());
31. workvo.setForfree(work.getForfree());
33. List<String> imagelist = **new** ArrayList<>();
34. String pictures=work.getPictures();
35. **if**(pictures!=**null**&&!"".equals(pictures)){
36. String[] temp = pictures.split(",");
37. **if**(temp.length>1||!temp[0].equals("")){
38. **for** (String image : temp) {
39. imagelist.add(work.getWorkmanid() + "/" + image);
40. }
41. }
42. }
43. workvo.setImagelist(imagelist);
45. workvo.setKeyword(work.getKeyword());
46. workvo.setScore(work.getScore());
47. workvo.setKeyword(work.getKeyword());
49. String workaddress=work.getWorkaddress();
50. **if**(workaddress!=**null**&&!"".equals(workaddress)){
51. workvo.setDladdress(work.getWorkmanid() + "/"+ workaddress);
52. }
54. User user=userService.getUser(work.getWorkmanid());
55. workvo.setMaker(user.getUsername());
56. **switch** (work.getWorktype()) {
57. **case** 1:
58. workvo.setType("化学");
59. **break**;
60. **case** 2:
61. workvo.setType("物理");
62. **break**;
63. **case** 3:
64. workvo.setType("生物");
65. **break**;
66. **default**:
67. workvo.setType(**null**);
68. }
69. **return** workvo;
70. }**else**{
71. System.out.println("函数调用错误");
72. **return** **null**;
73. }
74. }

*因为结果页只是给个查找结果，并不需要完整的展示实验的所有信息（这个应该在详情页），所以上面这个方法，假如显示的页面是结果页，图片只需要第一张图片，实验介绍也只取了前面的一部分字，后面用…省略，效果如下：*



3、效果图

效果如图5-10 所示

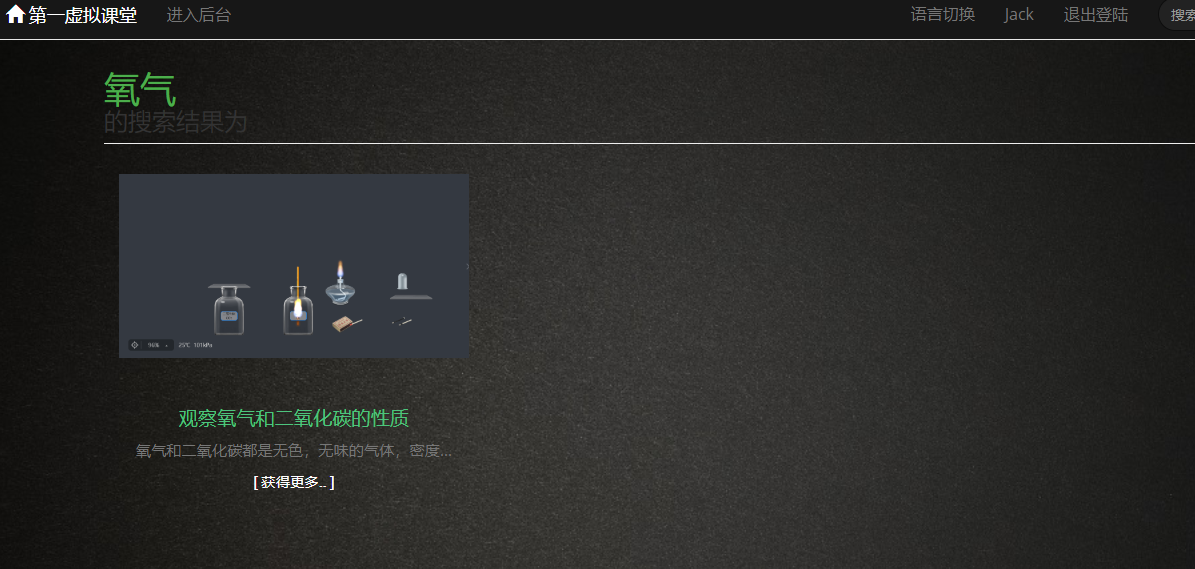


图5-10 搜索结果页

### 5.2.8 详情页模块

1、功能描述：展示实验的详细信息，包括标题，实验图片、描述，评论，以及提供实验的下载链接。

2、功能实现：详情页页面为workdetail.html。展示信息如标题、图片、描述和评论等在通过了后台查询后，都使用Thymleaf的${}标签展示。此外评论功能接收用户的打分以及评论详情后，通过在页面中保存着的一个隐藏的input标签（该标签保存了实验的主键信息），能够实现用户对特定实验的评价功能。下载功能主要通过http协议头的Content-Disposition字段实现，核心代码如下：

1. **<input** type="text" th:hidden="true" th:value="${workvo.dladdress}" id="dladdress"**/>**

4. **<div** th:replace="head :: headbar"**>**
5. **</div>**
7. **<div** class="about-sec"**>**
8. **<div** class="container"**>**
9. **<div** class="about-head"**>**
10. **<h2** th:text="${workvo.workname}"**>**实验名称**</h2>**
11. **</div>**
12. **<div** class="shiyan-grid"**>**
13. **<div** id="myCarousel" class="col-md-6 carousel slide shiyan"**>**
14. <!-- 轮播（Carousel）指标 -->
15. **<ol** class="carousel-indicators"**>**
17. **</ol>**
18. <!-- 轮播（Carousel）项目 -->
19. **<div** class="carousel-inner"**>**
20. **<div** th:if="${#lists.size(workvo.imagelist)>0}" th:each="image,iterStat:${workvo.imagelist}" th:class="${iterStat.first}?'lunboimage item active':'lunboimage item'" align="center"**>**
21. **<img** th:src="@{/showimage/\_\_${image}\_\_}"style="height: 100%;"**>**
22. **</div>**
23. **<div** class="item active lunboimage"  align="center" th:unless="${#lists.size(workvo.imagelist)>0}" **>**
24. **<img** th:src="@{/static/images/noimage.png}"**>**
25. **</div>**
26. **</div>**
27. <!-- 轮播（Carousel）导航 -->
28. **<a** class="carousel-control left" href="#myCarousel"
29. data-slide="prev"**>** **<span** \_ngcontent-c3="" aria-hidden="true" class="glyphicon glyphicon-chevron-left"**></span></a>**
30. **<a** class="carousel-control right" href="#myCarousel"
31. data-slide="next"**>** **<span** \_ngcontent-c3="" aria-hidden="true" class="glyphicon glyphicon-chevron-right"**></span></a>**
32. **</div>**
34. **<div** class="col-md-6 shiyan-info" style="height:320px;"**>**
35. **<div** class="panel panel-success"**>**
36. **<div** class="panel-heading" th:text="#{workdetail.introduction}"**></div>**
37. **<div** class="panel-body" th:text="${workvo.content}"**>**
38. Panel content
39. **</div>**
40. **</div>**
41. **<form** id="downloadform"**>**
42. **<h3** th:text="#{workdetail.keyword}+'：'+${workvo.keyword}" style="float:right;"**></h3>**
43. <!--登陆了的用户能够下载所有实验，没有登陆的只能下载免费的实验-->
44. **<button** style="position: absolute;bottom:0px;" type="button" class="btn btn-default" disabled="disabled" th:if="${workvo.dladdress==null||(session.currentUser==null&&workvo.forfree!=1)}"**>**未提供下载**</button>**
45. **<button** onclick="javascript:download()" style="position: absolute;bottom:0px;" type="button" class="btn btn-success" th:if="${(workvo.dladdress!=null&&session.currentUser!=null)||(workvo.dladdress!=null&&session.currentUser==null&&workvo.forfree==1)}"**>**[[#{workdetail.download}]]**</button>**
46. **</form>**
47. **</div>**
48. **<div** class="clearfix"**></div>**
49. **</div>**
50. **</div>**
51. **</div>**
52. **<div** class="news">
53. **<div** class="container">
54. <div class="col-md-8">
55. <h3 th:text="#{workdetail.comment}">评论</h3>
56. <form action="/assess" id="assessform" method="post">
57. <div class="form-group">
58. <label th:text="#{workdetail.scoring}">给实验打分</label>
59. <p><div id="layuitest"></div></p>
60. </div>
61. <div class="form-group">
62. <label th:text="#{workdetail.writecomments}">写下你的评论</label><span style="color: red;" id="tixing"></span>
63. <textarea class="form-control" rows="3" style="resize:none;" id="assesscontent" name="content"></textarea>
64. </div>
65. <input type="text" th:hidden="true" th:value="0" id="starvalue" name="starvalue"/>
66. <input type="text" th:hidden="true" th:value="${workvo.workid}" id="workid" name="workId"/>
67. <input type="text" th:hidden="true" th:if="${session.currentUser!=null}" th:value="${session.currentUser.id}" id="valuerid" name="valuerId"/>
68. <button type="button" class="btn btn-default submitbtn" onclick="javascript:starsubmit()">[[#{workdetail.submit}]]</button>
69. </form>
70. </div>
71. <div class="event-grids col-md-4">
72. <h3 th:text="#{workdetail.recentcomments}">最近的评论</h3>
73. <div class="event-grid" th:each="assess:${assesslist}">
74. <div class="date">
75. <h4 th:text="${#strings.substring(assess.assessdate,8,10)}">26</h4>
76. <span th:text="${#strings.substring(assess.assessdate,0,7)}">08/2012</span>
77. </div>
78. <div class="event-info">
79. <h5 style="color: #dd001b" th:text="${assess.star}+'分'">\*\*\*\*\*\*\*\*</h5>
80. <p th:text="${assess.content}" >评论内容</p>
81. <h5 class="accessman" th:text="${assess.assessername}">John.Mr</h5>
82. </div>
83. </div>
84. </div>
85. </div>
86. </div>
87. </body>
88. <script>
89. //打分
90. function starsubmit(){
91. var assesscontent=$("#assesscontent").val();
92. var starvalue=$("#starvalue").val();
93. debugger;
94. if(starvalue==0){
95. $("#tixing").text("[[#{workdetail.scoringtip}]]"); //国际化
96. }else if(assesscontent==""){
97. $("#tixing").text("[[#{workdetail.commenttip}]]");
98. }else{
99. $("#assessform").submit();
100. }
101. }
102. //下载
103. function download(){
104. var dladdress=$("#dladdress").val();
105. //debugger;
106. //使用post方法提交表单并下载
107. $("#downloadform").attr("action","/download?fileaddress="+dladdress);
108. $("#downloadform").attr("method","post").submit();
109. }
110. //打分设置
111. layui.use('rate', function(){
112. var rate = layui.rate;
113. //渲染
114. var ins1 = rate.render({
115. elem: '#layuitest' //绑定元素
116. ,choose: function (value) {
117. $("#starvalue").val(value);
118. }
119. });
120. });
121. </script>

*第一行为对应实验的下载地址，为下文的下载按钮服务。*

*第4行即引入导航条。*

*第7-40行展示实验的基本信息，实验图片*

*第41-46为一个from表单，通过from表单提交的方式，实现实验下载功能。其中，第44-45有两个按钮，根据登陆情况以及实验是否免费的情况确定显示哪一个。*

*第56-69行为又一个表单，实现评论功能。*

*第70-93行展示该实验的评论信息。*

*第96-108为打分的js方法，当点击评论提交按钮后就会触发该方法*

*第109-116行为下载的js方法，通过表单提交下载请求，当点击下载按钮时就会触发。*

*第119-130为layui框架的打分设置，根据用户选择的星星数目确定打多少分。*

3、效果图

效果如图5-11 所示



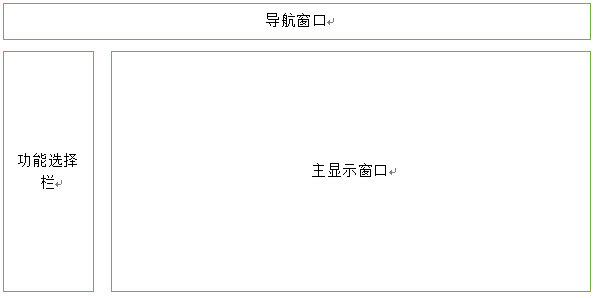
图5-11 实验详情图

## 5.3 用户后台功能模块实现

只有实验创作者才能够进入后台功能模块，故下面都是针对创作者而言：

所有的后台操作都在Controller包下的AdminController.java中完成

上面也说过，后台界面是如下的结构：



其中主显示窗口就是一个tab，使用的是layui的tab。

父窗口就是**admin/adminindex.html:**

1. <!DOCTYPE html**>**
2. **<html** xmlns:th="http://www.thymeleaf.org" xmlns="http://www.w3.org/1999/html"**>**
3. **<head>**
4. **<meta** charset="utf-8"**>**
5. **<meta** name="viewport" content="width=device-width, initial-scale=1, maximum-scale=1"**>**
6. **<title>** 后台管理 - Layui**</title>**
7. **<link** rel="stylesheet" th:href="@{/static/layui/css/layui.css}"**>**
8. **<script** th:src="@{/static/layui/layui.js}"**></script>**
9. **<script** th:src="@{/static/layui/jquery-3.3.1.min.js}"**></script>**
10. **</head>**
12. **<body** class="layui-layout-body"**>**
14. **<input** type="text" th:hidden="true" th:value="${session.currentUser.id}" id="userid"**>**
16. **<div** class="layui-layout layui-layout-admin"**>**
17. **<div** class="layui-header"**>**
18. **<div** class="layui-logo"**>**后台管理**</div>**
19. <!-- 头部区域（可配合layui已有的水平导航） -->
20. **<ul** class="layui-nav layui-layout-left"**>**
21. **<li** class="layui-nav-item"**><a** href="/"**>**首页**</a></li>**
22. **</ul>**
23. **<ul** class="layui-nav layui-layout-right"**>**
24. **<li** class="layui-nav-item"**>**
25. **<a** href="javascript:;" th:text="${session.currentUser.username}"**>**
26. **<img** src="http://t.cn/RCzsdCq" class="layui-nav-img"**>**
28. **</a>**
29. **<dl** class="layui-nav-child"**>**
30. **<dd><a** href="javascript:opentab('个人信息','userinfo');"**>**基本资料**</a></dd>**
31. **</dl>**
32. **</li>**
33. **<li** class="layui-nav-item"**><a** href="/logout"**>**退出登录**</a></li>**
34. **</ul>**
35. **</div>**
37. **<div** class="layui-side layui-bg-black"**>**
38. **<div** class="layui-side-scroll"**>**
39. <!-- 左侧导航区域（可配合layui已有的垂直导航） -->
40. **<ul** class="layui-nav layui-nav-tree"  lay-filter="test"**>**
41. **<li** class="layui-nav-item layui-nav-itemed"**>**
42. **<a** class="" href="javascript:opentab('个人信息','userinfo');"**>**个人信息**</a>**
43. **</li>**
44. **<li** class="layui-nav-item"**>**
45. **<a** href="javascript:;"**>**实验管理**</a>**
46. **<dl** class="layui-nav-child"**>**
47. **<dd><a** href="javascript:opentab('所有实验','userworks');"**>**所有实验**</a></dd>**
48. **<dd><a** href="javascript:opentab('创建实验','creatework');"**>**创建实验**</a></dd>**
49. **</dl>**
50. **</li>**
51. **</ul>**
52. **</div>**
53. **</div>**
55. **<div** class="layui-body"**>**
56. <!-- 内容主体区域 -->
57. **<div** style="padding: 15px;"**>**
58. **<div** class="layui-tab layui-tab-card" lay-allowClose="true" lay-filter="tabmanager"**>**
59. **<ul** class="layui-tab-title"**>**
61. **</ul>**
62. **<div** class="layui-tab-content" style="height: 100%;"**>**
64. **</div>**
65. **</div>**
66. **</div>**
67. **</div>**
69. **<div** class="layui-footer"**>**
70. © unity
71. **</div>**
72. **</div>**
74. **<script>**
75. var element;
76. layui.use('element', function(){
77. element=layui.element;
78. });
79. **</script>**
81. **<script>**
83. function opentab(text,url,param){
84. //假如参数为空，说明要传入的是当前用户id
85. if(param==undefined){
86. param=$("#userid").val();
87. }
88. element.tabAdd('tabmanager', {
89. title:text,
90. content:"**<iframe** frameborder=0 scrolling='auto' style='width:100%;height:450px'src='/admin/gethtml?pageName="+url+"¶m="+param+"'**></iframe>**",
91. id:'url'
92. });
93. abouttab();
94. }
95. **</script>**
97. **<script>**
98. //直接显示刚添加的tab,同时显示tab的标题和内容
99. function abouttab(){
100. if($('.layui-tab-item').hasClass('layui-show')){
101. $('.layui-tab-item').removeClass('layui-show');
102. }
103. if($("li[lay-id=url]").hasClass("layui-this")){
104. $("li[lay-id=url]").removeClass("layui-this");
105. }
106. $(".layui-tab-item:last").addClass("layui-show");
107. $("li[lay-id=url]:last").addClass("layui-this");
108. }
109. **</script>**
110. **</body>**
111. **</html>**

*58-65行就是一个tab容器，根据在功能选择栏点击的功能，然后显示对应的页面。下文说到的查看实验、实验创建、图片及实验上传其实就是一个子页面，显示在tab上。*

*第83-95就是在功能选择栏点击了之后会触发的方法，用到了layui框架的element.tabAdd方法，会到后台 /admin/gethtml 处调用返回子页面。如第47行的javascript:opentab('所有实验','userworks'); 就会打开子页面 userworks.html*

**Controller包下的AdminController.java:**

1. //根据名字打开特定的Tab
2. @RequestMapping(value = "/admin/gethtml",method = RequestMethod.GET)
3. **public** String getHtml(Model model,
4. @RequestParam(value="pageName")String name,
5. @RequestParam(value = "param")String param){
7. model.addAttribute("workmanid",Integer.parseInt(param));
9. **if**(name.equals("userinfo")){
10. //打开用户信息页面
11. User user=userMapper.selectByPrimaryKey(Integer.parseInt(param));
12. model.addAttribute("user",user);
13. }**else** **if**(name.equals("addimage")){
14. //打开新增图片页面
15. Integer workid=Integer.parseInt(param);
16. Work work= workServiceImpl.getWork(workid);
17. Integer workmanid=work.getWorkmanid();
18. ArrayList<String> images=**new** ArrayList<>();
19. String ss=work.getPictures();
20. **if**(ss!=**null**&&!ss.equals("")){
21. String[] temp=ss.split(",");
22. **for**(**int** i=0;i<temp.length;i++){
23. images.add(workmanid+"/"+temp[i]);
24. }
25. }
26. model.addAttribute("images",images);
27. model.addAttribute("workid",workid);
28. model.addAttribute("workmanid",work.getWorkmanid());
29. }**else** **if**(name.equals("workupload")){
30. //打开上传实验页面
31. Integer workid=Integer.parseInt(param);
32. Work work= workServiceImpl.getWork(workid);
33. model.addAttribute("workid",workid);
34. model.addAttribute("workname",work.getWorkname());
35. }
36. **return** "admin/"+name;
37. }

*根据传入的html文件名，打开admin文件夹下的对应的html页面。*

### 5.3.1 查看实验

1、功能描述：查看用户创建的所有实验，并且可以进行上传图片和实验的操作

2、功能实现：主要页面为userworks.html，所有的实验列表是通过html的<table>标签展示，而表格里的内容使用ajax请求从后端中获取。实验的类型字段本是整型值，所以表格在展示之前还通过了js来转换成字符类型。为了实现分页的目的，后端还使用了PageHelper插件以及返回了实验总数等参数。部分代码如下：

**userworks.html：**

1. **<body>**
2. **<table** id="worktable" lay-filter="worktabletool"**>**
3. **</table>**
4. **<input** type="text" id="workmanid" th:hidden="true" th:value="${workmanid}"**>**
5. **</body>**
7. <!--工具条-->
8. **<script** type="text/html" id="barDemo"**>**
9. **<a** class="layui-btn layui-btn-warm layui-btn-xs" lay-event="add"**>**实验上传**</a>**
10. <!-- <a class="layui-btn layui-btn-xs" lay-event="edit">编辑</a>    todo -->
11. **<a** class="layui-btn layui-btn-warm layui-btn-xs" lay-event="addimage"**>**新增图片**</a>**
12. <!-- <a class="layui-btn layui-btn-danger layui-btn-xs" lay-event="del">删除</a> todo-->
13. **</script>**
15. **<script** type="text/javascript"**>**
17. layui.use(['table','laypage','layer','element','laytpl'], function(){
18. var laypage=layui.laypage;
19. var table = layui.table;
20. var layer=layui.layer;
21. var element = layui.element;
22. var laytpl=layui.laytpl;
23. //第一个实例
24. table.render({
25. elem: '#worktable'
26. ,url: '/admin/getSomeoneWorks' //数据接口
27. ,where: { workmanid:$("#workmanid").val() } //接口参数
28. ,page: true //开启分页
29. ,method: 'get'
30. ,limit:10
31. ,limits:[5,10,15]
32. ,cols: [    //layui中方括号必须这么写（上下各一个，而不能写成"[["这样,否则thyleaf模板解析错误)
33. [ //表头
34. {field: 'id', title: 'ID', width:100,  fixed: 'left'}
35. ,{field: 'workname', title: '作品名称', width:100}
36. ,{field: 'keyword', title: '关键词', width:100}
37. ,{field: 'content', title: '介绍', width:100}
38. ,{field: 'worktype', title: '作品类型', width:100, templet: '#typeTpl'}
39. ,{field: 'option', title: '选项', width:180,toolbar: '#barDemo'}
40. ]
41. ]
42. });
44. //必须把on这个方法写在use方法里，否则on方法为未定义，固定的写法
45. table.on('tool(worktabletool)', function(obj){ //注：tool是工具条事件名，test是table原始容器的属性 lay-filter="对应的值"
46. var data = obj.data; //获得当前行数据
47. var layEvent = obj.event; //获得 lay-event 对应的值（也可以是表头的 event 参数对应的值）
48. var tr = obj.tr; //获得当前行 tr 的DOM对象
50. if(layEvent === 'del'){ //删除  //todo
51. layer.confirm('真的删除博客分类么', function(index){
52. //向服务端发送删除指令
53. $.get("/admin/delBlogType",{
54. id : data.id
55. },function(data1,textStatus){
56. //textStatus假如不是success，不执行回调函数,但是我的后台捕捉了异常，
57. //所以一定会成功执行这回调函数，故通过后台返回的Msg是否删除成功
58. if(data1.msg === '200'){
59. layer.msg("删除成功", {icon :1});
60. obj.del(); //删除对应行（tr）的DOM结构，并更新缓存
61. layer.close(index);
62. }else{
63. layer.msg("删除失败，该分类下可能有文章",{icon :2});
64. layer.close(index);
65. }
66. },"json");
67. });
68. }else if(layEvent === 'add'){  //添加
69. parent.opentab('上传实验','workupload',data.id);   //传入作品id
70. }else if((layEvent === 'addimage')){  //为作品新增图片
71. parent.opentab('增加图片','addimage',data.id);
72. }
73. });
74. });
75. **</script>**
77. <!--layui的模板引擎-->
78. **<script** type="text/html" id="typeTpl"**>**
79. {{#  if(d.worktype == 1){ }}
80. [[#{index.Chemistry}]]
81. {{#  } else if(d.worktype == 2){ }}
82. [[#{index.Physics}]]
83. {{#  }else { }}
84. [[#{index.Biology}]]
85. {{# } }}
86. **</script>**
87. **</html>**

*主要就是通过第2行的一个table显示内容。这里用到了layui框架的table。*

*第24-42行就是table的渲染方法，即从后台取到数据，并展示出来*

*第45-74行为table的选项栏的按钮的触发方法，目的是完成实验上传和新增图片的功能。*

*第78-86行为table作品类型 栏的转换方法，因为该字段为整型值，所以这里把他转换为字符串便于理解。*

**Controller包下AdminController.java:**

1. /\*\*
2. \* 获取某个作者的作品，有分页处理
3. \* @param manid
4. \* @param model
5. \* @param page
6. \* @param limit
7. \* @return
8. \*/
9. @RequestMapping(value = "/admin/getSomeoneWorks",method = RequestMethod.GET)
10. @ResponseBody
11. **public** Result showOnesWorks(@RequestParam("workmanid") Integer manid, Model model,
12. @RequestParam(value = "page",defaultValue = "1")**int** page,
13. @RequestParam(value = "limit",defaultValue = "5") **int** limit){
14. PageHelper.startPage(page,limit,"id asc");
15. Result result=**new** Result();
16. ArrayList<Work> list=(ArrayList) workServiceImpl.getsomeoneWorks(manid);
17. PageInfo<Work> datameta = **new** PageInfo<>(list);
18. List<Work> works=datameta.getList();
19. result.setCount((**int**)datameta.getTotal());
20. result.setData(works);
21. result.setCode(0);
22. **return** result;
23. }

3、效果图：

效果如图5-12 所示



图5-12 查看已创建的实验

### 5.3.2 实验创建

1、功能描述：创建一个新的实验，确定实验的标题，实验分类，实验的关键词以及填写实验的描述。

2、功能实现：主要的页面为creatework.html，填写完相关信息后，进行校检过程，确保没有空白项，然后调用底层数据库方法，将实验加入数据库中，但是此时实验图片和实验下载选项暂时为空，可以在后续步骤上传。

**Controller包下AdminController.java:**

1. /\*\*
2. \* 创建一个实验
3. \* @param work
4. \* @return
5. \*/
6. @RequestMapping(value = "/admin/addwork",method = RequestMethod.POST)
7. @ResponseBody
8. **public** Result addWork(@RequestBody Work work) **throws** Exception{
9. Result result=**new** Result();
10. **if**(workServiceImpl.createNewWork(work)==1){
11. **return** result;
12. }**else**{
13. result.setCode(400); //上传失败
14. }
15. **return** result;
16. }

3、效果图

效果如图5-13所示

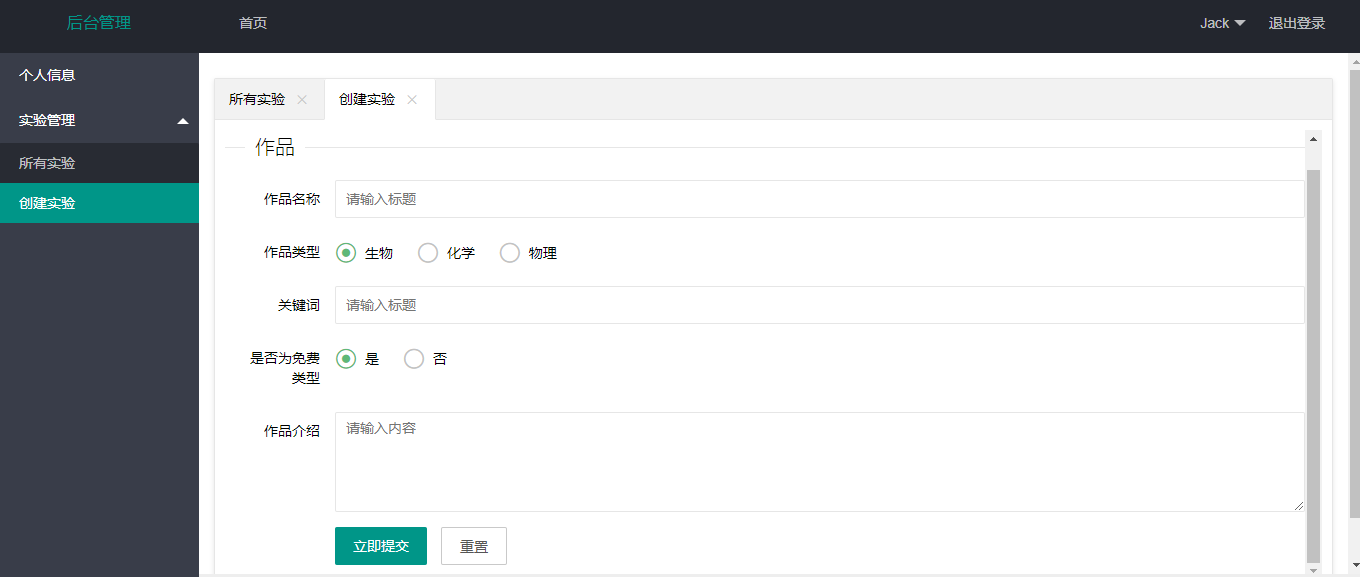


图5-13 创建实验

### 5.3.3 图片及实验上传

1、功能描述：在实验创建完成后，还必须上传实验以及用于展示用的实验图片。

2、功能实现：主要页面为addimage.html和workupload.html，无论是图片还是实验本质上都是文件上传，所以前端使用type属性为file的<input>标签上传文件，后端接收文件后，处理文件的文件名以及在数据库中保存文件的路径，完成上传过程。

**admin/addimage.html：**

1. **<input** type="text" id="workid" th:hidden="true" th:value="${workid}"**>**
2. **<input** type="text" id="workmanid" th:hidden="true" th:value="${workmanid}"**>**
4. **<fieldset** class="layui-elem-field layui-field-title" style="margin-top: 30px;"**>**
5. **<legend>**图片**</legend>**
6. **</fieldset>**
8. **<div** style="margin-bottom: 20px;"**>**
9. **<div**  th:each="image:${images}" class="layui-col-md4"**>**
10. **<div** class="showimages"**>**
11. **<img** th:src="@{/upload/\_\_${image}\_\_}" th:onclick="'delimage(\''+${image}+'\')' "**/>**
12. **</div>**
13. **</div>**
14. **</div>**
16. **<div** class="layui-col-md12"**>**
17. **<fieldset** class="layui-elem-field layui-field-title " style="margin-top: 30px;"**>**
18. **<legend>**上传图片**</legend>**
19. **</fieldset>**
21. **<div** class="layui-upload"**>**
22. **<button** type="button" class="layui-btn" id="test2"**>**选择图片**</button>**
23. **<blockquote** class="layui-elem-quote layui-quote-nm" style="margin-top: 10px;"**>**
24. 预览图：
25. **<div** class="layui-upload-list" id="preview"**></div>**
26. **</blockquote>**
27. **<button** class="layui-btn" id="toUpload"**>**
28. **<i** class="layui-icon"**>****</i>**上传
29. **</button>**
30. **</div>**
31. **</div>**
32. **</body>**
34. **<script>**
35. layui.use(['upload','layer'], function(){
36. var upload = layui.upload;
37. var layer=layui.layer;
39. var workid=$("#workid").val();
40. var workmanid=$("#workmanid").val();
42. //多图片上传
43. var uploaduploadInst=upload.render({
44. elem: '#test2'
45. ,url: '/admin/uploadimg/'
46. ,multiple: true
47. ,auto:false
48. ,bindAction:'#toUpload'
49. ,field:'files'
50. ,data:{workid:workid,workmanid:workmanid}
51. ,choose: function(obj){
52. //预读本地文件示例，不支持ie8
53. obj.preview(function(index, file, result){
54. $('#preview').append('**<img** src="'+ result +'" alt="'+ file.name +'" height="200" width="200" class="layui-upload-img"**>**')
55. });
56. }
57. ,allDone: function(obj){ //当文件全部被提交后，才触发
58. $('#preview').empty();   //删除预览图
59. layer.open({
60. title: '通知'
61. ,content: '上传成功'
62. });
63. },done: function(res){  //单个文件上传成功后触发
64. console.log("ok");
65. }
66. });
67. });
69. function delimage(img){
70. var layer=layui.layer;
71. //var s=img.split('/');
72. var workid=$("#workid").val();
73. layer.confirm("真的删除该图片吗？",function(index){
74. $.get("/admin/delimg",{
75. img : img,
76. workid:workid
77. },function(data1,textStatus){
78. //textStatus假如不是success，不执行回调函数
79. if(data1.code == 400){
80. layer.msg("删除失败，联系管理员",{icon :2});
81. layer.close(index);
82. }else {
83. layer.msg("删除成功", {icon :1});
84. layer.close(index);
85. }
86. },"json");
87. });
88. }

91. **</script>**

*第4行-14行 为已经上传的图片显示区域*

*第16-31行 为上传图片操作区域，可以选定多张图上传，有一个缩略图可以看*

*第35-67行 为上传图片的js方法*

*第69-88行 为删除图片的js方法，当在已上传图片区域点击对应图片，就会触发该方法。*

**admin/workupload.html：**

1. **<body** class="container"**>**
2. **<input** type="text" th:hidden="true" th:value="${workid}" id="w\_u\_workid"**/>**
4. **<br** **/>**
5. **<span** id="time"**></span>**
6. **<div** class="row" th:text="'实验名称：'+${workname}"**>**
7. **</div>**
8. **<div** class="row"**>**
9. **<input** class="layui-btn layui-btn-primary" type="file" name="file" **/><br** **/>**
10. **</div>**
11. **<div** class="row"**>**
12. **<div** id="progressBar" class="layui-progress layui-progress-big" lay-showPercent="yes" lay-filter="demo"**>**
13. **<div** class="layui-progress-bar layui-bg-green" lay-percent="0%"**></div>**
14. **</div>**
15. <!-- 显示上传速度 -->
16. **<div** id="showInfo" class="col-lg-2"**>**0KB/s**</div>**
17. **</div>**
18. <!-- 显示文件信息 -->
19. **<div** id="showFieInfo" class="row"**>**
20. **<label** name="upfileName"**></label><br** **/>**
21. **<label** name="upfileSize"**></label><br** **/>**
22. **<label** name="upfileType"**></label><br** **/>**
23. **</div>**
25. <!-- 上传时显示警告信息 -->
26. **<div** class="row" id="uploadingmsg"**>**
27. 上传时不要进行其他操作
28. **</div>**
30. **<div** class="row"**>**
31. **<input** class="layui-btn layui-btn-normal" type="button" name="upload" value="上传" **/>**
32. **<input** class="layui-btn layui-btn-warm" type="button" name="cancelUpload" value="取消" **/>**
33. **</div>**
34. **</body>**


38. **<script** type="text/javascript"**>**
39. var fileBtn = $("input[name=file]");          //上传的文件域
40. var processBar= $("#progressBar");
41. var uploadBtn=$("input[name=upload]");        //上传按钮
42. var canelBtn=$("input[name=cancelUpload]");   //取消按钮
43. var ot;         //上传开始时间
44. var oloaded;    //已上传文件大小
46. hideProgress(processBar);
47. $("#uploadingmsg").hide();
48. //文件域绑定监听方法
49. fileBtn.change(function() {
50. var fileObj = fileBtn.get(0).files[0]; //js获取文件对象
51. if (fileObj) {
52. var fileSize = getSize(fileObj.size);
53. $("label[name=upfileName]").text('文件名：' + fileObj.name);
54. $("label[name=upfileSize]").text('文件大小：' + fileSize);
55. $("label[name=upfileType]").text('文件类型：' + fileObj.type);
56. uploadBtn.attr('disabled', false);
57. }
58. });
60. //进度条依赖于element模块
61. layui.use(['element','layer'], function(){
62. var element = layui.element;
63. var layer = layui.layer;
64. // 上传文件按钮点击的时候
65. uploadBtn.click(function(){
66. // 进度条归零
67. element.progress('demo', '0%');  //这里使用的时layui的进度条
68. // 上传按钮禁用
69. $(this).attr('disabled', true);
70. // 上传文件
71. uploadFile();
72. });
74. function uploadFile(){
75. var url ="/admin/workupload";
76. var fileObj = fileBtn.get(0).files[0];
77. if(fileObj==null){
78. alert("请选择文件");
79. return;
80. }
81. // 进度条显示
82. showProgress(processBar);
83. $("#uploadingmsg").show();
84. // FormData 对象
85. var form = new FormData();
87. form.append('files', fileObj); // 文件对象
88. form.append("workid",$("#w\_u\_workid").val());//其他需要上传的表单内容
90. // XMLHttpRequest 对象
91. var xhr = new XMLHttpRequest();
92. //true为异步处理
93. xhr.open('post', url, true);
94. //上传开始执行方法
95. xhr.onloadstart = function() {
96. console.log('开始上传')
97. ot = new Date().getTime();   //设置上传开始时间
98. oloaded = 0;//已上传的文件大小为0
99. };
101. xhr.upload.addEventListener('progress', progressFunction, false);
102. xhr.addEventListener("load", uploadComplete, false);
103. xhr.addEventListener("error", uploadFailed, false);
104. xhr.addEventListener("abort", uploadCanceled, false);
105. xhr.send(form);
107. function progressFunction(evt) {
108. if (evt.lengthComputable) {
109. var completePercent = Math.round(evt.loaded / evt.total \* 100)+"%";
111. element.progress('demo', completePercent);
113. var time = $("#time");
114. var nt = new Date().getTime();     //获取当前时间
115. var pertime = (nt-ot)/1000;        //计算出上次调用该方法时到现在的时间差，单位为s
116. ot = new Date().getTime();          //重新赋值时间，用于下次计算
118. var perload = evt.loaded - oloaded; //计算该分段上传的文件大小，单位b
119. oloaded = evt.loaded;               //重新赋值已上传文件大小
121. //上传速度计算
122. var speed = perload/pertime;//单位b/s
123. var bspeed = speed;
124. var units = 'b/s';//单位名称
125. if(speed/1024**>**1){
126. speedspeed = speed/1024;
127. units = 'k/s';
128. }
129. if(speed/1024**>**1){
130. speedspeed = speed/1024;
131. units = 'M/s';
132. }
133. speedspeed = speed.toFixed(1);
134. //剩余时间
135. var resttime = ((evt.total-evt.loaded)/bspeed).toFixed(1);
136. $("#showInfo").html(speed+units+'，剩余时间：'+resttime+'s');
137. }
138. }
140. //上传成功后回调
141. function uploadComplete(result) {
142. uploadBtn.attr('disabled', false);
143. $("#uploadingmsg").hide();
144. layer.open({
145. title: '通知'
146. ,content: '上传成功'
147. });
148. };
150. //上传失败回调
151. function uploadFailed(result) {
152. console.log('上传失败' + result.msg);
153. $("#uploadingmsg").hide();
154. layer.open({
155. title: '通知'
156. ,content: '上传失败，请联系管理员'
157. });
158. }
160. //终止上传
161. function cancelUpload() {
162. xhr.abort();
163. }
165. //上传取消后回调
166. function uploadCanceled(evt) {
167. console.log('上传取消,上传被用户取消或者浏览器断开连接:' + evt.target.responseText);
168. }
170. canelBtn.click(function(){
171. uploadBtn.attr('disabled', false);
172. cancelUpload();
173. })
174. }
176. });

179. function getSize(size) {
180. var fileSize = '0KB';
181. if (size **>** 1024 \* 1024) {
182. fileSize = (Math.round(size / (1024 \* 1024))).toString() + 'MB';
183. } else {
184. fileSize = (Math.round(size / 1024)).toString() + 'KB';
185. }
186. return fileSize;
187. }
188. function showProgress(processBar) {
189. processBar.parent().show();
190. }
191. function hideProgress(processBar) {
192. processBar.parent().hide();
193. }
194. function setProgress(element,w) {
195. element.progress('demo', w+"%");
196. }
197. **</script>**

*这里上传实验的进度条使用的是layui框架的进度条*

3、效果图

效果如图5-14和5-15所示

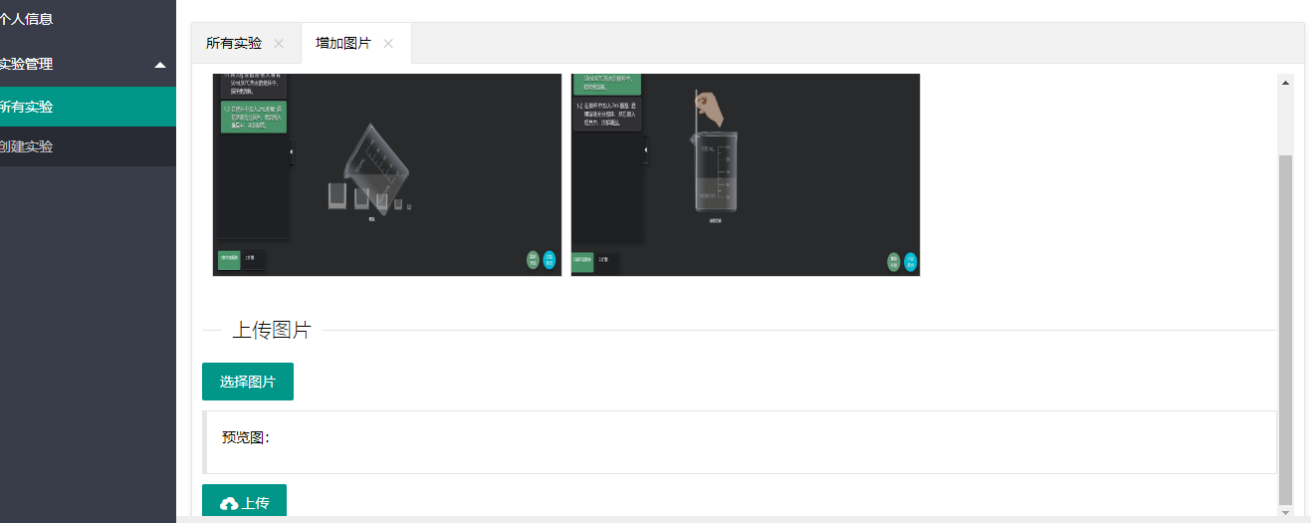


图5-14 上传图片

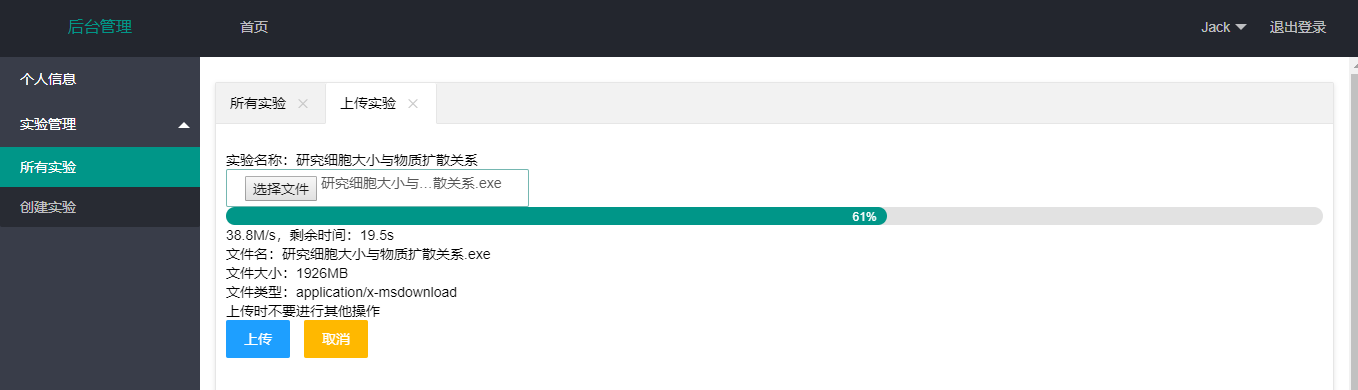


图5-15 上传实验

## 5.4 管理员后台功能模块实现

管理员是网站的保留用户，无法通过注册成为管理员，管理员的职责是维护网站，如添加公告和使用案例，查看所有用户和某个用户的已上传实验。

这里的后台布局和 5.3用户后台功能模块实现一致，故此处不再赘述。

这里的主要后台操作都在controller包下的RootController.java文件下

### 5.4.1 公告和使用案例模块

1、功能描述：管理员可以添加和删除用于首页展示的公告和使用案例。

2、功能实现：页面为cases.html和news.html。前端先通过js验证输入的公告和使用案例的标题和内容以及图片是否为空，然后提交到数据库中。点击删除按钮时将公告或者使用案例的主键传入后台，数据库中将数据或图片删除。

3、效果图：



图5-16 使用案例和公告模块

### 5.4.2 查看用户模块

1、功能描述：查看网站所有已注册的用户的信息，以及他们上传的实验信息。

2、功能实现：页面为allusers.html。数据库直接查询用户表的所有记录，前端js将用户类型字段由整形转为字符串，然后使用表格展示。

3、效果图



图5-17 查看用户模块

# 总结

兴趣是学习的路上最好的老师，而能够进行实验则是我们收获兴趣的关键。课堂实验不仅仅是熟悉操作流程，更重要的是通过实验现象看透实验本质问题，从而达到消化知识的目的。但是在有些地方，因为实验场地问题或者实验材料的问题，又或者是因为实验的危险性问题，导致无法进行实验，仅仅是通过他人的结论来理解消化知识，使得学习效果大打折扣。虚拟实验完美的解决了上面的所有问题，虚拟性和交互性使得实验变得安全又方便。但是我们又缺少这么一个平台，以至于一个简单的实验还要在无边际的因特网中苦苦搜寻，虚拟实验网站平台应运而生。

在通过指导老师的悉心指导下和相关书籍的帮助下，在该虚拟网站开发过程中具体完成的主要工作包括：

1、编写了该网站的需求分析和总体设计。

2、实现了用户注册与登录，实验搜索，实验上传和下载功能。

3、其中主要集中完成实验相关部分，包括了实验的创建，实验的上传和实验图片的上传，以及实验的关键词模糊查询功能。还有就是后台的管理功能，包括了使用案例和网站公告的展示，用户信息查询和实验信息查询。

经过了这次毕业设计，给我提供了一个学以致用的机会，使得我的动手能力大大增强。但是同时也暴露出了一些问题，比如对数据安全的不够重视以及对于计算机软件的鲁棒性考虑不够充分，相信这也是我未来要加强学习的部分。

# 数据库建表语句

CREATE TABLE `assess` (

`id` int(11) NOT NULL AUTO\_INCREMENT,

`valuerid` int(11) DEFAULT NULL COMMENT '评价者的id',

`workid` int(11) DEFAULT NULL COMMENT '评价的作品id',

`content` varchar(500) DEFAULT NULL COMMENT '评价的内容',

`star` int(11) DEFAULT NULL COMMENT '评价的星级',

`assessdate` varchar(40) DEFAULT NULL COMMENT '评价的时间',

`assessername` varchar(40) DEFAULT NULL COMMENT '评价人昵称',

PRIMARY KEY (`id`)

) ENGINE=InnoDB AUTO\_INCREMENT=46 DEFAULT CHARSET=utf8;

CREATE TABLE `news` (

`id` int(11) NOT NULL AUTO\_INCREMENT,

`newsname` varchar(100) NOT NULL COMMENT '公告标题',

`content` varchar(500) NOT NULL COMMENT '公告内容',

`ctime` varchar(20) NOT NULL COMMENT '公告创建时间',

PRIMARY KEY (`id`)

) ENGINE=InnoDB AUTO\_INCREMENT=8 DEFAULT CHARSET=utf8;

CREATE TABLE `usecases` (

`id` int(11) NOT NULL AUTO\_INCREMENT,

`casename` varchar(100) NOT NULL COMMENT '案例标题',

`content` varchar(500) NOT NULL COMMENT '案例内容',

`image` varchar(50) NOT NULL,

PRIMARY KEY (`id`)

) ENGINE=InnoDB AUTO\_INCREMENT=8 DEFAULT CHARSET=utf8;

CREATE TABLE `user` (

`id` int(11) NOT NULL AUTO\_INCREMENT,

`username` varchar(20) DEFAULT NULL COMMENT '用户名',

`email` varchar(40) DEFAULT NULL COMMENT '用户邮箱',

`tel` varchar(20) DEFAULT NULL COMMENT '用户电话',

`registrationtime` varchar(40) DEFAULT NULL COMMENT '注册时间',

`usertype` int(11) DEFAULT NULL COMMENT '用户类型 0普通用户 1制作者',

`password` varchar(50) NOT NULL COMMENT '用户密码',

PRIMARY KEY (`id`) USING BTREE

) ENGINE=InnoDB AUTO\_INCREMENT=7 DEFAULT CHARSET=utf8 ROW\_FORMAT=DYNAMIC;

CREATE TABLE `work` (

`id` int(11) NOT NULL AUTO\_INCREMENT,

`workname` varchar(50) DEFAULT NULL COMMENT '作品名称',

`workmanid` int(11) DEFAULT NULL COMMENT '作品作者的id',

`content` varchar(500) DEFAULT NULL COMMENT '作品内容介绍',

`pictures` varchar(1000) DEFAULT NULL COMMENT '作品图片',

`keyword` varchar(50) DEFAULT NULL COMMENT '作品关键字',

`worktype` int(3) NOT NULL COMMENT '实验类型 1化学 2物理 3生物',

`score` mediumtext COMMENT '获得的评价总分',

`workaddress` varchar(200) DEFAULT NULL COMMENT '实验保存地址',

`forfree` int(3) DEFAULT '1' COMMENT '1为免费下载，0为其他',

PRIMARY KEY (`id`)

) ENGINE=InnoDB AUTO\_INCREMENT=87 DEFAULT CHARSET=utf8;

CREATE TABLE `complain` (

`id` int(11) NOT NULL AUTO\_INCREMENT,

`manid` int(11) DEFAULT NULL COMMENT '投诉者的id',

`content` varchar(500) DEFAULT NULL COMMENT '投诉的内容',

`complaintime` varchar(40) DEFAULT NULL COMMENT '评价的时间',

`manname` varchar(40) DEFAULT NULL COMMENT '投诉人昵称',

PRIMARY KEY (`id`)

) ENGINE=InnoDB AUTO\_INCREMENT=0 DEFAULT CHARSET=utf8;

管理员用户需要预先在数据库中设置，插入语句为：

INSERT INTO `user`(`id`, `username`, `email`, `tel`, `registrationtime`, `usertype`, `password`) VALUES (1, '邱文捷', '932964177@qq.com', '13113176571', '2019-2-23', 3, '90fe32d8ba4135db6265c70aa48892fb');

比较关键的是email、password、usertype字段，其中usertype必须为3，password为经过MD5加密后的字符，可以通过下面的方法获得该字符串，email则可以自定义。

1. **package** qiutest;
2. **import** org.apache.shiro.crypto.hash.Md5Hash;
3. **public** **class** Mytest {
4. **public** **static** **void** main(String[] args) {
5. Md5Hash md5Hash=**new** Md5Hash("123","unity");
6. System.out.println(md5Hash.toString());
7. }
8. }

把“123”改成需要的密码，运行该方法即可。