同济大学计算机科学与技术系

数据库系统原理课程设计报告

科研资讯推送系统

作业项目 科研资讯推送系统

学　　号 1751740

姓　　名 刘鲲

专　　业 计算机科学与技术

授课老师 向阳 教授

日　　期 2020/05/18

目录

1. 引言

1.1. 背景

对于科研工作者来说，科研资讯的实时获取十分重要。而目前存在的科研资讯获取软件、平台和渠道或多或少存在问题，无法较好地满足科研工作者科研资讯的推送需求。

在对于现有科研资讯获取软件、平台和渠道的调查和分析的基础上，取长补短，明确新系统的设计目标和功能， 以满足科研工作者的需求。

同时，作为数据库系统原理的课程设计课，合适地运用在《数据库原理导论》课上的所学知识，完成一个具有实际意义的数据库系统。

1.2. 要求

1.2.1. 设计要求

（1）按照软件工程中的软件生命周期来设计应用系统,撰写报告要求有如下内容:

1）问题定义；2）可行性分析；3）需求分析；4）总体设计；5）详细设计；6）编码与单元调式；7）综合测试；8）软件维护（详见数据库课程设计模板）

（2）强调数据库设计，要使用 E-R 图设计概念模型；要设计逻辑模型和物理模型

（3）要考虑规范化和实际应用需要，一般要求达到三范式(3NF)

（4）完整性设计，关系模型的三类完整性约束条件在设计的过程中是必须考虑的，数据之间的关联应详细说明，要求使用 DBMS 对联系进行适当定义和编辑。对有些统计数据可使用触发器。

（5）安全性设计，数据库的安全性是至关重要的，建议为系统设置用户管理功能，系统的用户至少分为两级：系统管理员和一般用户。不同级别的用户可操作的功能是不一样的。

（6）系统体系结构设计采用 B/S 模式。

（7）应用程序功能设计，应用系统的基本功能应根据实际目标来设定，通常有增、删、改、查、打印、备份、恢复、密钥等功能。

（8）在课程设计期间按要求完成设计任务，本学期第 17 周五课设任务结束。

（9）课设结束时要求提交如下内容：课设报告、设计系统的源代码文件、桌面录象演示三类文件，都刻在光盘上。助教收齐统一提交。同时在 17 周进行系统远程网上运行检查。

1.2.2. 报告要求

1）应用系统程序应独立完成，报告和程序功能完整，设计方法合理，用户界面较好，系统运行正常。

2）课程设计提交的设计报告请按照软件工程的要求与格式书写。

3）不少于 2 万字（ (需求分析、概念模型部分不得少于 1 万字) ）;

4）装订顺序：封面、目录、正文、参考文献（ 以同济大学本科毕业设计模版为准）;

5）提交报告需同时提供报告电子版,以及设计系统的源代码文件和桌面录象演示文件，都刻在光盘上。助教收齐统一提交。同时要求远程网上运行检查。

6）设计报告严禁抄袭。

1.3. 目标

1.3.1. 总体目标

加深对数据库系统、程序设计语言和软件工程的理论知识的理解和应用水平。

通过设计实际的数据库系统应用课题，进一步熟悉数据库应用系统的操作技术，提高学生分析问题和解决问题的能力，强化学生的动手能力。

1.3.2. 具体目标

1）运用数据库设计理论设计一个较完善的有实际意义的数据库；

2）掌握目前流行数据库管理系统 MySQL 或 SQL Server 或 ORACLE 的应用与开发技术；

3）利用某种高级语言，为数据库开发相应的应用程序，形成完整的数据库应用系统；

2. 需求分析

2.1. 业务调查

科研资讯对于一个科研人员来说非常重要。

科研资讯一般分为几种：

A. 领域跟进
B. 科研人员的讨论、观点
C. 讲座、大会

D. 文章截稿时间

以上的信息都具有时效性。换言之，作为一个科研人员必须一直跟进。

其中第一点是最重要的。一个能够个性化定制并管理以上所有信息的平台是科研人员需要的。

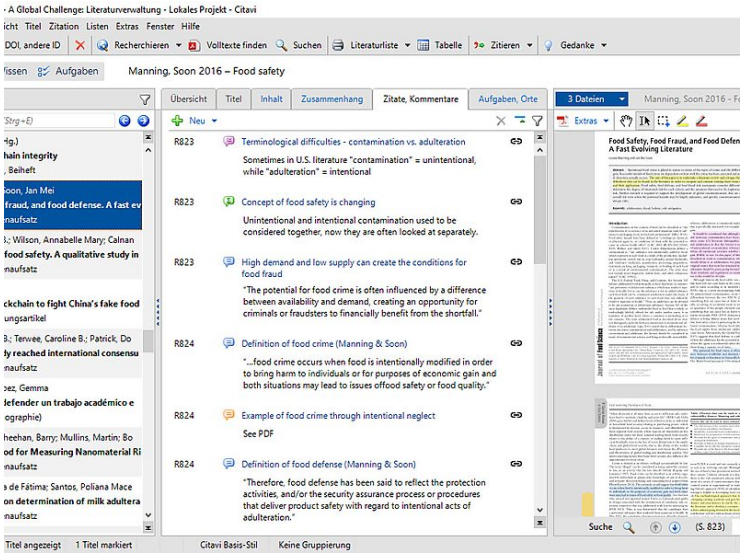
目前论文管理软件有：

A. Zotero
B. Citavi
C. Mendeley
D. Endnote

以笔者使用的 Citavi 为例，它具有很齐全的文献识别功能，导入文献的时候能够识别标题、作者；

具有注释标注、知识管理、计划任务、团队协作云协作功能；具有在线搜索、添加引文、文献管理、文档摘录（PDF、Word 等）；

导出时，支持边写边引、创建参考书目、创建汇编和列表并导出数据。

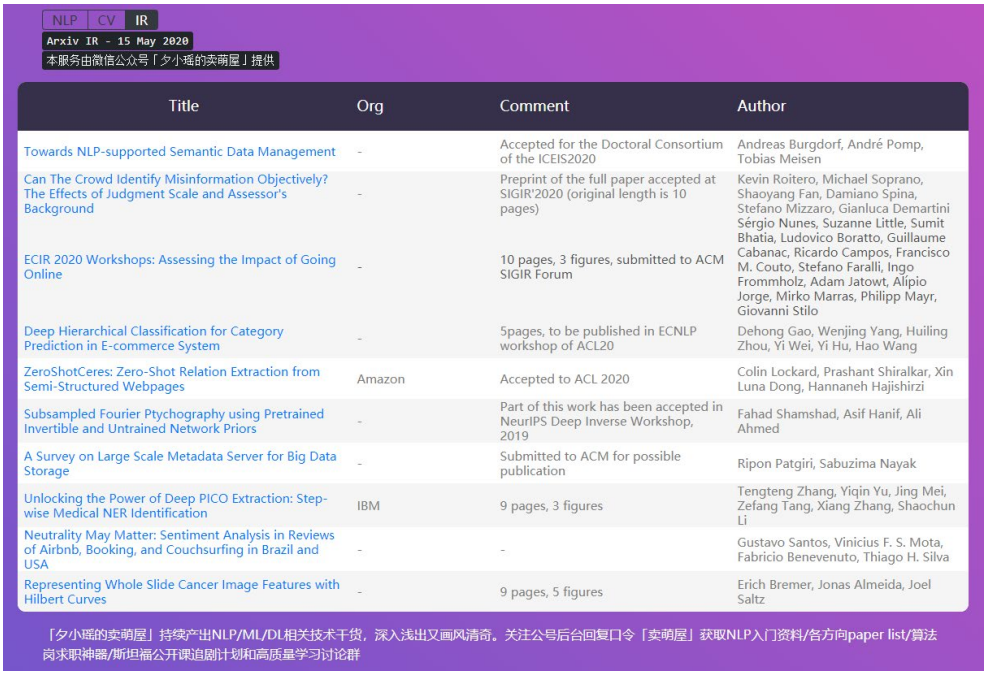


其它的文献管理软件大同小异，但它们都具有一个特征：主动性。用户需要自己去查询、添加论文。

并且，它们处理的信息并不具有时效性，使用场景多为写作科研阶段，并不适合跟进领域的研究进展。

2.1.1. 新论文推送

一般来说，新论文多出于 arXiv 等网站，需要自己去检索，如有人就在做这样的推送：



NLP CV IR

Arxiv IR - 15 May 2020

本服务由微信公众号「夕小瑶的卖萌屋」提供

Title	Org	Comment	Author
Towards NLP-supported Semantic Data Management	-	Accepted for the Doctoral Consortium of the ICEIS2020	Andreas Burgdorf, André Pomp, Tobias Meisen
Can The Crowd Identify Misinformation Objectively? The Effects of Judgment Scale and Assessor's Background	-	Preprint of the full paper accepted at SIGIR'2020 (original length is 10 pages)	Kevin Roitero, Michael Soprano, Shaoyang Fan, Damiano Spina, Stefano Mizzaro, Gianluca Demartini
ECIR 2020 Workshops: Assessing the Impact of Going Online	-	10 pages, 3 figures, submitted to ACM SIGIR Forum	Sérgio Nunes, Suzanne Little, Sumit Bhatia, Ludovico Boratto, Guillaume Cabanac, Ricardo Campos, Francisco M. Couto, Stefano Faralli, Ingo Frommholz, Adam Jatowt, Alípio Jorge, Mirko Marras, Philipp Mayr, Giovanni Stilo
Deep Hierarchical Classification for Category Prediction in E-commerce System	-	5pages, to be published in ECNLP workshop of ACL20	Dehong Gao, Wenjing Yang, Huiling Zhou, Yi Wei, Yi Hu, Hao Wang
ZeroShotCeres: Zero-Shot Relation Extraction from Semi-Structured Webpages	Amazon	Accepted to ACL 2020	Colin Lockard, Prashant Shiralkar, Xin Luna Dong, Hannaneh Hajishirzi
Subsampled Fourier Ptychography using Pretrained Invertible and Untrained Network Priors	-	Part of this work has been accepted in NeurIPS Deep Inverse Workshop, 2019	Fahad Shamshad, Asif Hanif, Ali Ahmed
A Survey on Large Scale Metadata Server for Big Data Storage	-	Submitted to ACM for possible publication	Ripon Patgiri, Sabuzima Nayak
Unlocking the Power of Deep PICO Extraction: Step-wise Medical NER Identification	IBM	9 pages, 3 figures	Tengteng Zhang, Yiqin Yu, Jing Mei, Zefang Tang, Xiang Zhang, Shaochun Li
Neutrality May Matter: Sentiment Analysis in Reviews of Airbnb, Booking, and Couchsurfing in Brazil and USA	-	-	Gustavo Santos, Vinicius F. S. Mota, Fabricio Benevenuto, Thiago H. Silva
Representing Whole Slide Cancer Image Features with Hilbert Curves	-	9 pages, 5 figures	Erich Bremer, Jonas Almeida, Joel Saltz

「夕小瑶的卖萌屋」持续产出NLP/ML/DL相关技术干货，深入浅出又画风清奇。关注公号后台回复口令「卖萌屋」获取NLP入门资料/各方向paper list/算法岗求职神器/斯坦福公开课追剧计划和高质量学习讨论群

以上是一个简单的静态网页。

以下则是知乎上对于 arXiv 论文的推送号：

语音/音频每日论文速递[11.20]

arXiv **arXivDaily**，公众号: arXiv每日学术速递/计算机视觉/机器学习/划水

> 同步公众号(arXiv每日论文速递)，欢迎关注，感谢支持哦~

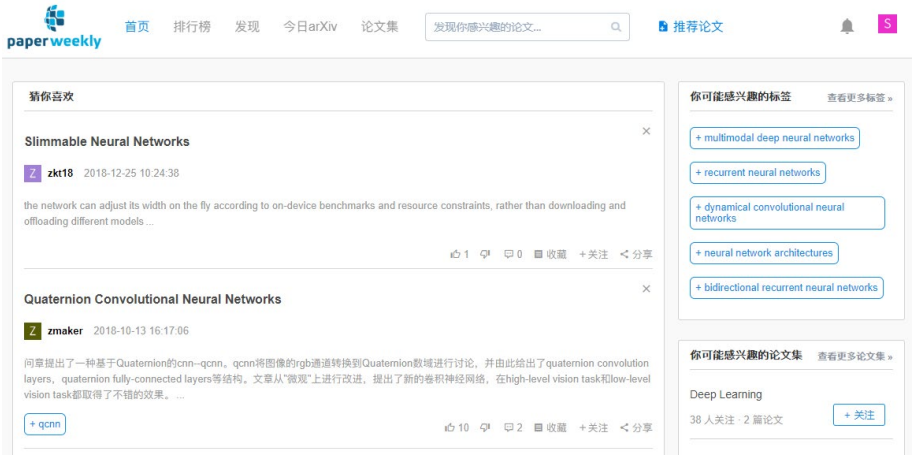
cs.SD 方向，今日共计9篇

【1】 Alternating Between Spectral and Spatial Estimation for Speech Separation and Enhancement
标题：谱估计和空间估计的交替用于语音分离和增强
作者： Zhong-Qiu Wang, John R. Hershey
链接：arxiv.org/abs/1911.0795...

【2】 Improving Universal Sound Separation Using Sound Classification
标题：利用声音分类改进通用声音分离
作者： Efthymios Tzinis, Daniel P. W. Ellis
链接：arxiv.org/abs/1911.0795...

固然科研人员可以每日定时登上这些平台查看，但是如果研究领域在一段时间内相对固定，每次检索的条件基本相同，那么就不需要每天做这样的操作；并且，如果论文来源不止一个，每次检索也需要切换平台，并不是很方便。

目前国内有一个专门针对计算机科学的文献推荐网站 paperweekly，但它只做计算机科学，并且它只有“推荐”而无“推送”。



它运作的模式是用户推荐论文并写上评语，根据推荐程度对相关领域进行推荐。

但这仅是“推荐”，将“不受欢迎”的论文对用户屏蔽，从科研角度来说失去了对当前领域完整面貌的把握；

并且容易造成马太效应，个别论文因为初期讨论度较高，挤压了其它论文的热度，

这种权重式的论文推荐方法不好。但它社区式的讨论是非常可取的。

2.1.2. 科研实体的社交账号

这是一个比较重要的信息来源。在国外的 Twitter，在国内的知乎，是科研人员集中并活跃发表观点的社交网络。

如下图，是清华大学计算机系教授刘致远的知乎账号：

如下图，是人工智能知名研究人员吴恩达的 Twitter 账号：

同时，一些公司和科研机构，也会公布它们的一些近期研究进展。如下图，是旷视科技的 Twitter 账号：

在国内，还有微信公众号这一重要的途径。如下图，是量子位的微信公众号：

这些是十分重要的信息来源，应当被利用起来。但是它们分散在各个地方；并且这些平台往往是社交平台，也就是说有很多与科研无关的资讯。为了避免“信息轰炸”，提高科研人员获取资讯的效率，我们应当将这些资讯隔离出来，统一到一个一站式平台。

2.1.3. 学术活动

学术活动一般为会议、讲座等。比起人为地主动记录会议投稿的截止时间，通过一个平台统一展示和提醒更保险，更利于统筹安排。

同时一些有一定价值但不太有人了解的会议和讲座，尤其是新办会议和讲座，需要一个宣传平台。

如下图，www.allconferences.com 是集中了所有会议的平台：

2.1.4. 总结

目前对于科研资讯的获取，存在以下痛点：

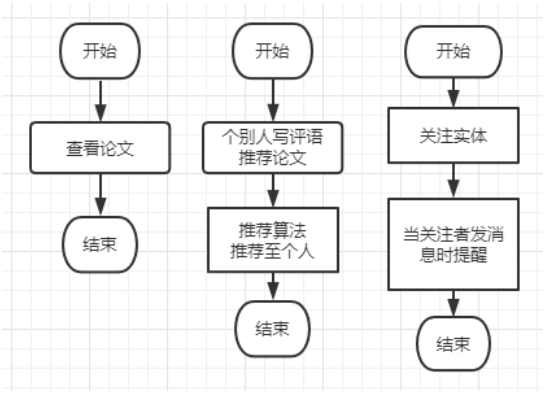
A. 绝大多数的科研资讯，都需要科研人员主动搜寻，并且分散在不同的平台，统一管理有一定难度
B. 对于科研实体，它们有发布、展示自己科研进展的需求，并参与进一步的讨论，但分散在不同的社交平台，没有打通和集中
C. 新晋会议、讲座主办方有宣传学术活动的需求，但学术圈子太小，前期宣传无门

对于以上痛点，对项目作如下分析：

A. 市面上的文献管理软件，已都较为成熟，故该项目不应该在这方面参与竞争
B. 文献检索也有很方便的 Google Scholar 等，不需要再造一个新轮子
C. 其次，不需要再建一个社交平台满足科研交流的需求，而应该想通如何打通不同社交平台的科研资讯
D. 资讯的推送是定制化的，可以适应不同的领域或跨领域，不会拘泥于一个领域

2.2. 原系统业务流程分析

前面分析的多种科研资讯获取模式，大多数是如下的三种方式：



第一个流程对应的是一些只有简单展示论文的平台。

这种平台需要自己打开，然后去点击自己感兴趣的领域，无法做到推送；而且每次查看论文的时候，都需要打开平台以及选择感兴趣的领域，这一步完全可以省略。

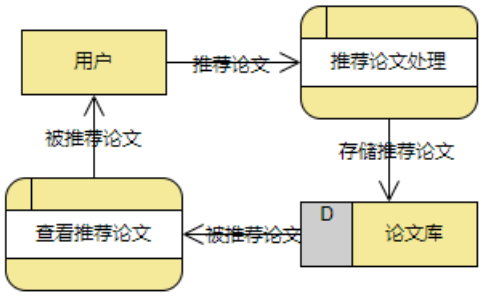
第二个流程对应的是 paperweekly 这样具有推荐功能的平台。

上文说到，这种推荐功能是“中心化”的，带有“偏见”的，而一般认为，一个科研人员应当有自己判断论文好坏的能力，而不能依赖于学术大牛的推荐。

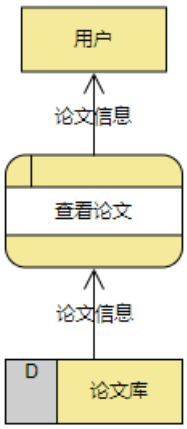
第三个流程对应的是社交平台。

科研资讯散落在各个平台，如果能把它们集中起来，做到一站式科研资讯推送，可以减轻用户原平台的“信息轰炸”。社交平台的主体并非科研资讯，这样就能集中用户的精力应对关键信息。

2.3. 原系统数据流程分析



这是对于推荐系统的数据流图。可以看到是一个闭环。用户推荐论文，进入系统论文库；然后论文库将推荐论文推送给用户。论文的推荐完全依赖于其它用户的推荐，作为科研人员获取的信息是有偏见的，故它适合作为一个交流平台和记录平台，而非一个第一手科研资讯的推送平台。



这是对于一般论文查看网站的数据流图。可以看到系统只有简单的存储论文的功能，而用户只能主动去特定网站查阅，并不是十分方便。

2.4. 新系统的目标（含新系统的功能需求、性能需求、输入输出需求）

2.4.1. 功能需求

提供科研资讯推送的定制化服务，打通不同社交平台的科研资讯，实现每日科研资讯推送，并且集中在一站内。

2.4.2. 性能需求

考虑到并非是电子商城、即时聊天软件这种对于实时性要求较高的系统，新系统需要做到响应时间粒度秒级以内；

考虑到科研人员并非很多，故新系统需做到并发响应数 100 人/秒即可；

由于学生无法租用大容量的服务器和数据库，故其性能完全受限于设备的性能；对于阿里云，有学生机可以租，型号为 ecs.n4.small，规格参数如下：

vCPU	1
内存（GiB）	**2.0**
本地存储（GiB）	None
网络带宽能力（出/入）（Gbit/s）	**0.5**
网络收发包能力（出+入）（万 PPS）	5
支持 IPv6	No
多队列	1
弹性网卡（包括一块主网卡）	2
单块弹性网卡的私有 IP	2

我们主要关注内存大小、本地存储和网络带宽能力这三项。一般来说，普通的私人服务器用不到 0.2Gbit/s 以上的带宽，而内存和本地存储则是越大越好，随着规模的扩增而扩增；并且阿里云有很方便的服务器迁移，所以要扩容加钱就是。

总体说来，租用的学生云服务器足以满足初期的性能需求。

精度

说明对该软件的输入、输出数据精度的要求，可能包括传输过程中的精度。

A. 输入数据精度：字符串类型和整数类型，无小数类型；字符串最长 256 字节，整数最大 20 位，也就是 Java 类型中的 Long，与 MySQL 的 BIGINT 类型相对应，作为主码；
B. 输出数据精度：字符串类型和整数类型，无小数类型；字符串最长 256 字节，整数最大 20 位，也就是 Java 类型中的 Long，与 MySQL 的 BIGINT 类型相对应，作为主码；
C. 传输过程精度：以 json 格式传输，字符串类型和整数类型，无小数类型；字符串最长 256 字节，整数最大 20 位，也就是 Java 类型中的 Long，与 MySQL 的 BIGINT 类型相对应，作为主码；

时间特性要求

说明对于该软件的时间特性要求。

响应时间：3 秒以内

更新处理时间：1 秒以内

数据的转换和传送时间：1 秒以内

灵活性

说明对该软件的灵活性的要求，即当需求发生某些变化时，该软件对这些变化的适应能力。

操作方式上的变化：

当使用不同权限的账号登录后，系统提供的功能不一样，这要求系统具有权限管理功能；而开源 JAVA 框架中，已有 Shiro 这个广泛使用的开源包，可以很方便地做到权限管理和用户认证。

运行环境的变化：

当更新数据库的时候，要求热更新，即服务不能停；开发环境在 Windows 下，运行环境在 Linux 下，要求两者没有区别；而 Java 具有一次编译，到处运行的特性，可以满足这一点。一般现在工业界的做法，也是在 Windows 下完成开发，因为有丰富和功能强大的 IDE，然后再上线 Linux 服务器。

同其他软件的接口的变化：

预留对接主流软件、社交网站的接口，方便拓展和扩大业务；同时我们需要注意设计系统的时候有无做好接口，设计是否合理；而现在的社交网站如 Twitter 一般都会提供开发者接口，我们需要去适应即可。

精度和有效时限的变化：

在并发数较高的情况下，允许响应时间增加 1s；对于一些论文的标题过长，超过了数据库的存储，允许截断；截断的原因在于在数据库里使用较长的字符串得不偿失，也影响性能。

2.4.3. 输入输出需求

解释各输入输出数据类型，并逐项说明其媒体、格式、数值范围、精度等。对软件的数据输出及必须标明的控制输出量进行解释并举例，包括对硬拷贝报告（正常结果输出、状态输出及异常输出）以及图形或显示报告的描述。

对于服务器来说，其输入输出是 HTTP 协议的请求和应答。因为 HTTP 中，POST 和 GET 都是 TCP 协议，没有本质上的区别，故这里不对数据输入和输出作严格区分。

输入数据格式

输入数据在前端，一般有两种形式。

一种是表单形式；在新系统中，有较多的填表操作，这些表单的传输方式有两种主流

方式，具体见下文的介绍。

另一种是参数形式，呈现在 HTTP 的 header，在 url 后，如“www.baidu.com?username=liukun&password=111111”等。这种方式比较危险，没有保密性。

POST，也就是服务器端返回的结果，一共有三种：

值	描述
application/x-www-form-urlencoded	在发送前编码所有字符（默认）
multipart/form-data	不对字符编码。在使用包含文件上传控件的表单时，必须使用该值。
text/plain	空格转换为 "+" 加号，但不对特殊字符编码。

application/x-www-form-urlencoded

最常见的 POST 提交数据的方式。浏览器的原生 <form> 表单，如果不设置 enctype 属性，那么最终就会以 application/x-www-form-urlencoded 方式提交数据。

```
<form action="form_action.asp" enctype="text/plain">
  <p>First name: <input type="text" name="fname" /></p>
  <p>Last name: <input type="text" name="lname" /></p>
  <input type="submit" value="Submit" />
</form>
```

此时 Form 提交的请求数据，抓包时（或者 Chrome 打开开发者模式查看）看到的请求会是这样的内容（无关的请求头已省略）：

```
POST http://www.example.com HTTP/1.1
Content-Type: application/x-www-form-urlencoded;charset=utf-8
title=test&sub%5B%5D=1&sub%5B%5D=2&sub%5B%5D=3
```

multipart/form-data

一个常见的 POST 数据提交的方式。我们使用表单上传文件时，必须让 <form> 表单的 enctype 等于 multipart/form-data。Content-Type 里指明了数据是以 multipart/form-data 来编码，本次请求的 boundary 是什么内容。消息主体里按照字段个数又分为多个结构类似的部分，每部分都是以 --boundary 开始，紧接着是内容描述信息，然后是回车，最后是字段具体内容（文本或二进制）。如果传输的是文件，还要包含文件名和文件类型信息。消息主体最后以 --boundary-- 标示结束。

application/json

实际上，application/json 这个 Content-Type 是作为响应头。实际上，HTTP 中 POST

和 GET 其实都是 TCP 协议，故也可以把它作为请求头，用来告诉服务端消息主体是序列化后的 JSON 字符串。由于 JSON 规范的流行，除了低版本 IE 之外的各大浏览器都原生支持 JSON.stringify，服务端语言也都有处理 JSON 的函数，使用 JSON 不会遇上什么麻烦。

JSON 格式支持比键值对复杂得多的结构化数据，对于新系统来说，常常有这样的表格化数据，十分方便。Google 的 AngularJS 中的 Ajax 功能，默认就是提交 JSON 字符串。例如下面这段代码：

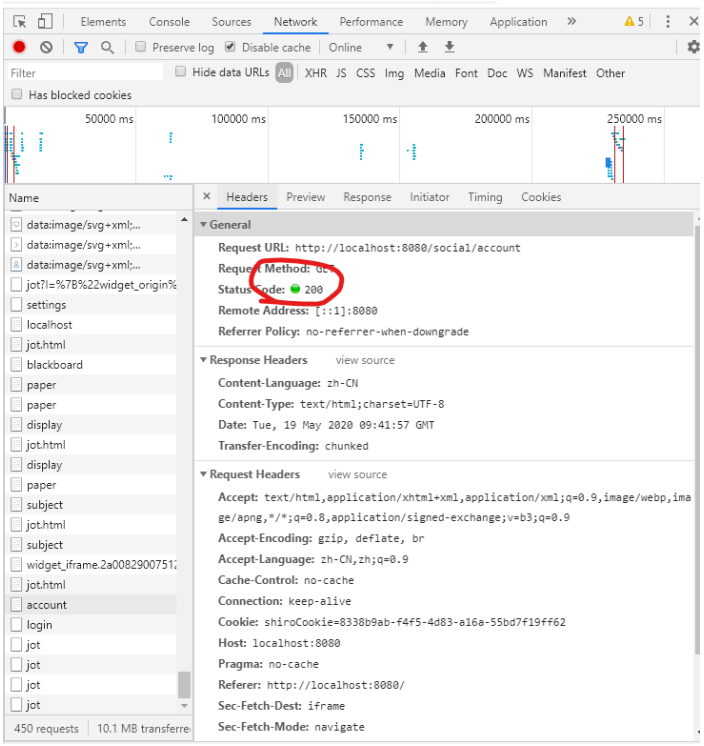
```
JSvar data = {'title':'test', 'sub' : [1,2,3]};
$http.post(url, data).success(function(result) {
    ...
});
```

最终发送的请求是：

```
POST http://www.example.com HTTP/1.1
Content-Type: application/json;charset=utf-8
{"title":"test","sub":[1,2,3]}
```

这种方案，可以方便的提交复杂的结构化数据，特别适合 RESTful 的接口。各大抓包工具如 Chrome 自带的开发者工具、Firebug、Fiddler，都会以树形结构展示 JSON 数据，非常友好。

对于 HTTP 来说，有许多状态码。Chrom 打开开发者模式即可查看，如图：



图中的 status code=200，表明数据传输正常。HTTP 的状态码如下：

(1) **正常输出**

200 OK 客户端请求成功

(2) **异常输出**

301 Moved Permanently 请求永久重定向

302 Moved Temporarily 请求临时重定向

304 Not Modified 文件未修改，可以直接使用缓存的文件。

400 Bad Request 由于客户端请求有语法错误，不能被服务器所理解。

401 Unauthorized 请求未经授权。这个状态代码必须和 WWW-Authenticate 报头域一起使用

403 Forbidden 服务器收到请求，但是拒绝提供服务。服务器通常会在响应正文中给出不提供服务的原因

404 Not Found 请求的资源不存在，例如，输入了错误的 URL

500 Internal Server Error 服务器发生不可预期的错误，导致无法完成客户端的请求。

503 Service Unavailable 服务器当前不能够处理客户端的请求，在一段时间之后，服务器可能会恢复正常。

在开发过程中，一般是打开 Chrome 的开发者模式，查看具体的 HTTP 报文是否正常发送，而正常与否，就依赖于这些状态码。

3. 系统分析

3.1. 新系统的子系统划分

本项目使用 Spring 进行开发，故先介绍 Spring 的子系统划分。在 Spring 的子系统划分的基础上，介绍新系统功能的子系统划分。

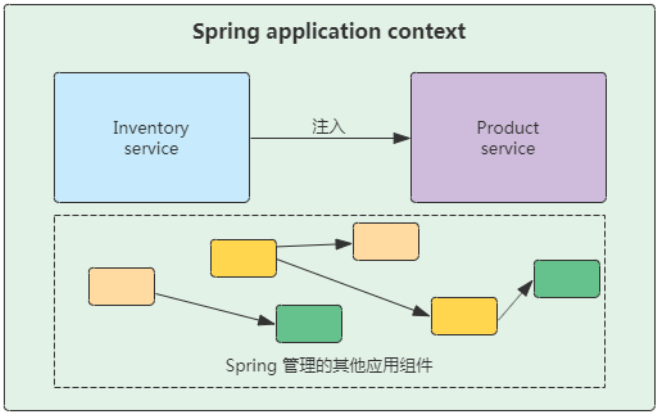
3.1.1. Spring 的子系统划分

Spring 的核心是一个容器，通常称为 Spring 应用程序上下文，用于创建和管理应用程序组件。这些组件（或 bean）在 Spring 应用程序上下文中连接在一起以构成一个完整的应用程序，就像将砖、灰浆、木材、钉子、管道和电线绑在一起以组成房屋。

将 bean 连接在一起的行为是基于一种称为 依赖注入（DI）的模式。依赖项注入的应用程序不是由组件自身创建和维护它们依赖的其他 bean 的生命周期，而是依赖于单独的实体（容器）来创建和维护所有组件，并将这些组件注入需要它们的 bean。通常通过构造函数参数或属性访问器方法完成此操作。

例如，假设在应用程序的许多组件中，要处理两个组件：

inventory service（用于获取库存级别）和 product service（用于提供基本产品信息）。product service 取决于 inventory service，以便能够提供有关产品的完整信息。



我认为，依赖注入是一个非常天才和伟大的发明创造，大大减少了开发过程中繁琐的主动注入过程，减少了耦合和依赖。

3.1.2. Spring Framework

也就是我们经常说的 spring 框架，包括了 ioc 依赖注入，Context 上下文、bean 管理、springmvc 等众多功能模块，其它 spring 项目比如 spring boot 也会依赖 spring 框架。

3.1.3. Spring Boot

Spring Boot 是由 Pivotal 团队提供的全新框架，其设计目的是用来简化新 Spring 应用的初始搭建以及开发过程。该框架使用了特定的方式来进行配置，从而使开发人员不再需要定义样板化的配置。Spring Boot 其实不是什么新的框架，它默认配置了很多框架的使用方式，就像 maven 整合了所有的 jar 包，Spring Boot 整合了所有的框架（不知道这样比喻是否合适）。

3.1.4. Spring Data

是一个数据访问及操作的工具集，封装了多种数据源的操作能力，包括：jdbc、Redis、MongoDB 等。使用 Spring Data，能简化对数据库的访问，加强项目的维护能力。

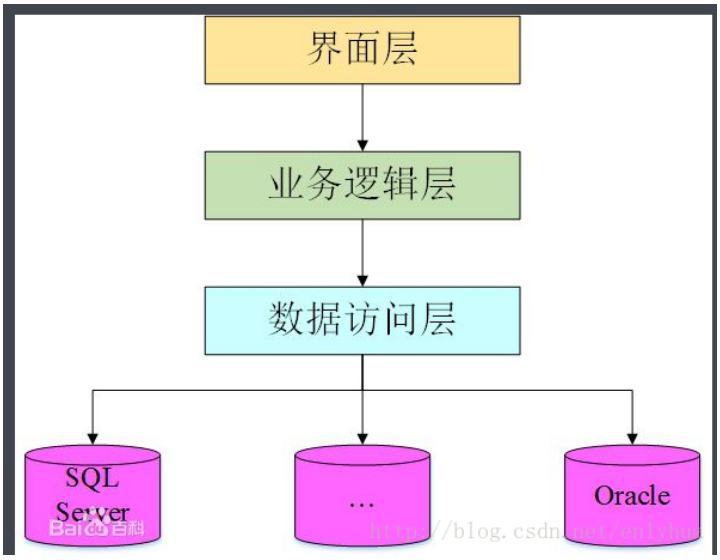
3.1.5. Spring Security

Spring Security 主要用于快速构建安全的应用程序和服务，在 Spring Boot 和 Spring Security OAuth2 的基础上，可以快速实现常见安全模型，如单点登录，令牌中继和令牌交换。比较常见的是 oauth2 授权机制和 jwt 认证方式。oauth2 是一种授权机制，规定了完备的授权、认证流程。JWT 全称是 JSON Web Token，是一种把认证信息包含在 token 中的认证实现，oauth2 授权机制中就可以应用 jwt 来作为认证的具体实现方法。

本课设使用 OAuth2 认证方式。

以上的都是框架有的，这里开始是需要自己实现的。

一般来说，主流的软件项目的子系统设计如下：



对于不同的框架，有不同的系统分割方法。以下给出新系统的分割方法：

3.1.6. Entity

此子系统实现所有的实体类。所谓实体，就是对应数据库里的一张表。比如 User 作为一个实体，对应数据库里的 user 表。而 User 的私有成员，就分别与数据库里的字段一一对应。

当然，也有不理想的情况。比如有些字段在 User 中不会存到数据库中；再比如有些数据库数据涉及安全，比如密码，就不能全部送给前端等等。这时候还有封装这一手法可以使用。根据阿里的 Java 开发手册，还有这些传输过程的封装：

A. DO（ Data Object）：与数据库表结构一一对应，通过 DAO 层向上传输数据源对象。
B. DTO（ Data Transfer Object）：数据传输对象，Service 或 Manager 向外传输的对象。
C. BO（ Business Object）：业务对象。 由 Service 层输出的封装业务逻辑的对象。
D. AO（ Application Object）：应用对象。 在 Web 层与 Service 层之间抽象的复用对象模型，极为贴近展示层，复用度不高。
E. VO（ View Object）：显示层对象，通常是 Web 向模板渲染引擎层传输的对象。

3.1.7. Service

此子系统实现所有的实体类对应的具体业务。一般说来，业务逻辑层中的模块包含了系统所需要的所有功能上的算法和计算过程，并与数据访问层和表现层交互。服务层就是相当于中间类的作用，中间的工厂类提供了另一个通用放任接口让调用者可以使用接口暴露的方法，而无需关注架构或底层发生的怎样的变化，服务层的原理和这个非常类似，只不过它将工厂模式应用到更高层面的抽象之上。

3.1.8. Controller

此子系统实现所有的来自网页的访问请求。它是用户和系统之间交流的桥梁，它一方面为用户提供了交互的工具，另一方面也为显示和提交数据实现了一定的逻辑，以便协调用户和系统的操作。

3.1.9. Dao

此子系统实现对于数据库的访问。数据访问层是一个代码类库，提供访问位于持久化容器中数据的功能，在分层设计中，所有从介质化读取数据或写入数据的工作都属于这一层的任务。

3.1.10. 前端资源

Static

实现所有页面的静态资源，如 html；以及样式表 Css。

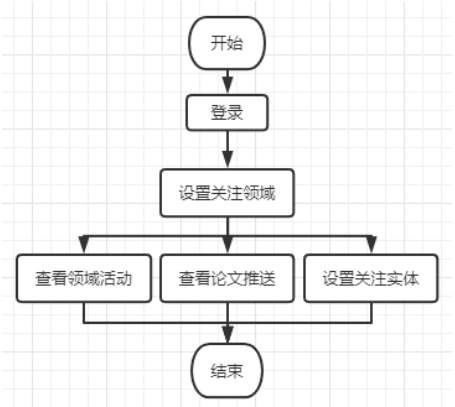
Js

这里是 JavaScript 文件，实现动态资源的加载。在新系统里，它一般用来完成以下任务：

A. 嵌入动态文本于 HTML 页面
B. 对浏览器事件作出响应
C. 读写 HTML 元素
D. 在数据被提交到服务器之前验证数据
E. 检测访客的浏览器信息
F. 控制 cookies，包括创建和修改等

对于前端这一部分，不是新系统的重点，故不花过多时间设计，而是使用 Bootstrap 这个框架完成。

3.2. 新系统的业务流程



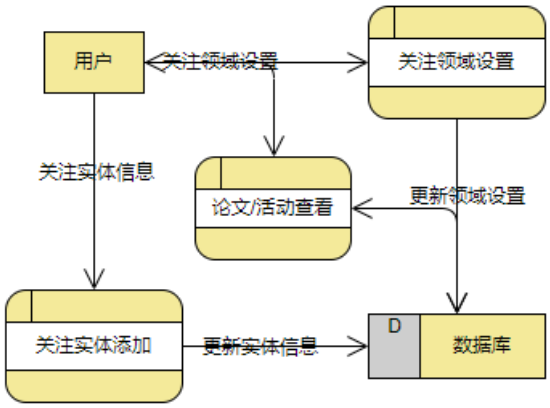
新系统业务流程说明：

用户登录后，设置好关注领域，就可以查看领域活动、论文推送；设置关注实体后，可以看到他们在社交网站上的发言、文章等。

“查看领域活动“、”查看领域推送“、“设置关注实体”和“设置领域关注”其实都在导航栏里。

在查看活动中，还有根据关键字搜索；而所谓的“设置”，包括增加、删除和修改。

3.3. 新系统的数据流程



上图的数据流图，涵盖了新系统的主要功能：领域设置、论文及科研活动查看，以及对于科研实体的关注。

3.4. 新系统的数据字典

3.4.1. user

字段名	数据类型	允许为空	PK	字段说明
id	bigint(20)	NO	自增	主键
account	varchar(255)	YES		账号名
create_by	bigint(20)	YES		创建人
create_time	datetime	YES		创建时间/注册时间
email	varchar(255)	YES		电子邮件
phone	Varchar(20)	YES		电话
depid	bigint(20	YES		部门 id
modify_by	bigint(20)	YES		最后更新人
modify_time	datetime	YES		最后更新时间
name	varchar(255)	YES		用户名
password	varchar(255)	YES		密码
roleid	varchar(255)	YES		角色 id
sex	int(11)	YES		性别
status	int(11)	YES		账号状态

说明：

可以看到，有些重要的字段比如名字等允许为空，虽然逻辑上不对，但这是为了现实实际运行效率考虑。在非主键上使用 Not Null 大大影响插入效率。所以实际工业界一般不设置 Not Null。

同样，还有一个关系不建议设置：外键。外键大大影响插入、查询效率。工业界也一般不使用外键，而是通过其它方法实现一对一、一对多、多对多映射。在 Hibernate 中，是通过注释实现。本系统是通过 Spring Data JPA 实现外键映射，它本质是 Hibernate 的封装。

另外，可以看到有四个字段：create_by, create_time, modify_by, modify_time，这也是工业界的习惯，在表内留下创建和修改信息方便写系统日志。

对一些字段作出更进一步说明：

角色 id：

新系统的设计是为管理员和用户设置不同的角色，而对角色设置不同的权限。这和 MySQL 的角色权限管理是差不多的，也就是给角色赋权，而不是具体用户，方便管理。

3.4.2. paper

字段名	数据类型	允许为空	PK	字段说明
id	bigint(20)	NO	自增	主键
author	varchar(512)	YES		作者
comment	varchar(255)	YES		备注
create_by	bigint(20)	YES		创建人
create_time	datetime	YES		创建时间/注册时间
modify_by	bigint(20)	YES		最后更新人
modify_time	datetime	YES		最后更新时间
org	varchar(255)	YES		所属组织
title	varchar(255)	YES		文章标题
url	varchar(255)	YES		链接

3.4.3. subject

字段名	数据类型	允许为空	PK	字段说明
id	bigint(20)	NO	自增	主键
create_by	bigint(20)	YES		创建人
create_time	datetime	YES		创建时间/注册时间
modify_by	bigint(20)	YES		最后更新人
modify_time	datetime	YES		最后更新时间
name	varchar(64)	YES		领域名字

pid	bigint(20)	YES		上级领域名字

3.4.4. social_account

字段名	数据类型	允许为空	PK	字段说明
id	bigint(20)	NO	自增	主键
create_by	bigint(20)	YES		创建人
create_time	datetime	YES		创建时间/注册时间
modify_by	bigint(20)	YES		最后更新人
modify_time	datetime	YES		最后更新时间
username	varchar(32)	YES		用户名
website	varchar(32)	YES		账号所在网站名

3.4.5. activity

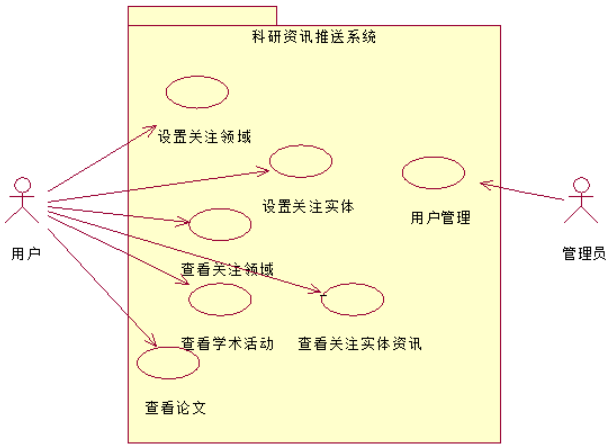
字段名	数据类型	允许为空	PK	字段说明
id	bigint(20)	NO	自增	主键名
begin_time	datetime	YES		活动开始时间
create_by	bigint(20)	YES		创建人
create_time	datetime	YES		创建时间/注册时间
end_time	datetime	YES		活动结束时间
modify_by	bigint(20)	YES		最后更新人
modify_time	datetime	YES		最后更新时间
name	varchar(255)	YES		活动名
url	varchar(255)	YES		活动链接

3.4.6. Role

字段名	数据类型	允许为空	PK	字段说明
id	bigint(20)	NO	自增	主键
create_by	bigint(20)	YES		创建人
create_time	datetime	YES		创建时间/注册时间
deptid	bigint(20)	YES		部门 id
modify_by	bigint(20)	YES		最后更新人
modify_time	datetime	YES		最后更新时间
name	varchar(255)	YES		名字
num	int(11)	YES		个数
pid	bigint(20)	YES		父角色 id
tips	varchar(255)	YES		提示

4. UML 设计

4.1. 系统用例图



说明：

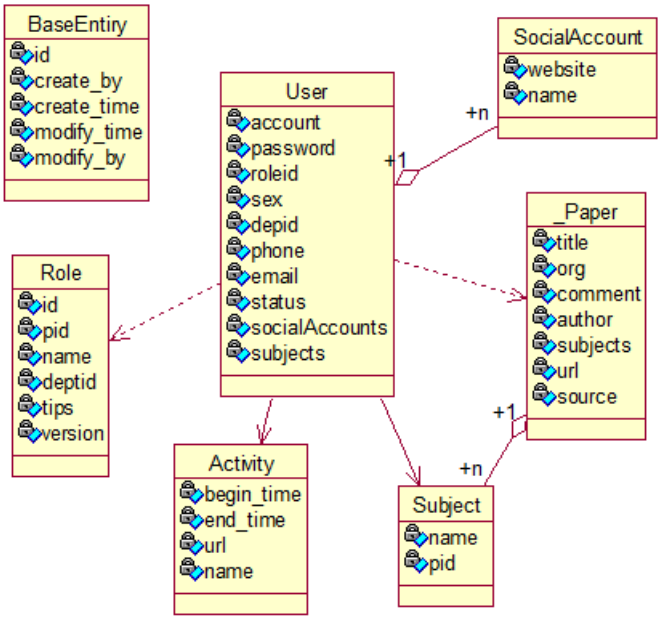
用户有以下用例：

设置关注领域；设置关注实体；查看关注领域；查看学术活动；查看关注实体资讯；查看论文；

管理员有以下实例：

用户管理

4.2. 类图



4.2.1. BaseEntity

一切实体的父类，也就是图中所有其它的类都是 BaseEntity 泛化而来。**图中是为了美观没画泛化的箭头。**

可以看到 BaseEntity 有修改时间，是强制要求所有类的新建修改都要记录时间。

Id 是一切类的唯一标识，也即唯一主键。数据类型为 long，对应 MySQL 的 bigint 数据类型。这里不选取其它数据类型作为主键，原因还是效率和开发难度。

MySQL 对于主键会建立唯一索引，而数据类型的索引效率比字符串高；而有一个基类的 BaseEntity 可以节省大量代码。

一个 Entity，对应数据库里的一个表。

4.2.2. User

对应数据库设计中的 user 表。

成员名	数据类型	字段说明
id	Long	主键
account	String	账号名
create_by	Long	创建人
create_time	Date	创建时间/注册时间
email	String	电子邮件
phone	String	电话
depid	Long	部门 id
modify_by	Long	最后更新人
modify_time	Date	最后更新时间

name	String	用户名
password	String	密码
roleid	String	角色 id
sex	Int	性别
status	Int	账号状态
socialAccounts	Set<SocialAccount>	关注的社交账号
subjects	Set<Subject>	关注的学科

对于 socialAccounts 和 subjects，这里设置这两个成员实际上是为了后续的多对多设计，这使用到了 Spring Data JPA 的特性，这将在代码设计一块提及。

其它的其实和数据字典的差不了多少，上文提及新系统没有浮点类型，只有整形和字符串，故涉及到的对于 JDBC 和 Java 类型之间的对应关系如下：

JDBC 类型	JAVA 类型
VARCHAR	String
LONGVARCHAR	String
INTEGER	int
DATETIME	Date

4.2.3. Subject

对应数据库设计中的 subject 表。

成员名	数据类型	成员说明
id	Long	主键
create_by	Long	创建人
create_time	Date	创建时间/注册时间
modify_by	Long	最后更新人
modify_time	Date	最后更新时间
name	String	领域名字
pid	Long	上级领域名字

4.2.4. Activity

对应数据库设计中的 activity 表。

成员名	数据类型	成员说明
id	Long	主键名
begin_time	Date	活动开始时间
create_by	Long	创建人
create_time	Date	创建时间/注册时间

end_time	Date	活动结束时间
modify_by	Long	最后更新人
modify_time	Date	最后更新时间
name	String	活动名
url	String	活动链接

4.2.5. Paper

对应数据库设计中的 paper 表。

字段名	数据类型	字段说明
id	Long	主键
author	String	作者
comment	String	备注
create_by	Long	创建人
create_time	Date	创建时间/注册时间
modify_by	Long	最后更新人
modify_time	Date	最后更新时间
org	String	所属组织
title	String	文章标题
url	String	链接

4.2.6. SocialAccount

对应数据库设计中的 account 表。

成员名	数据类型	成员说明
id	Long	主键
create_by	Long	创建人
create_time	Date	创建时间/注册时间
modify_by	Long	最后更新人
modify_time	Date	最后更新时间
username	String	用户名
website	String	账号所在网站名

4.2.7. Role

成员名	数据类型	字段说明
id	Long	主键
create_by	Long	创建人
create_time	Date	创建时间/注册时间

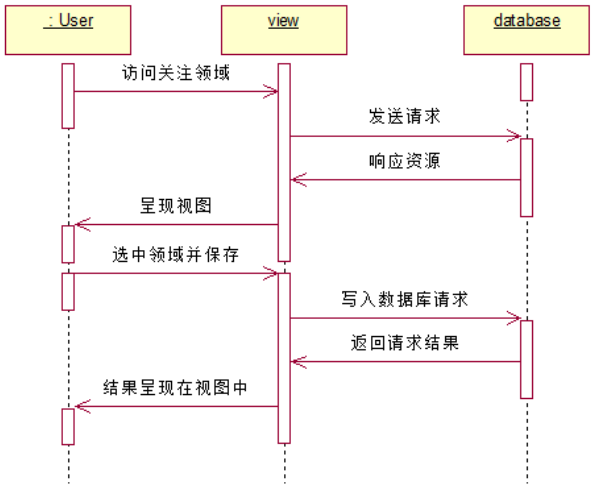
deptid	Long	部门 id
modify_by	Long	最后更新人
modify_time	Date	最后更新时间
name	String	名字
num	int	个数
pid	Long	父角色 id
tips	String	提示

在类的设计中，可以看到没有对成员方法的设计，原因在于现在工业界一般为了做到高内聚低耦合，所有相关的业务代码都和实体解耦，所以除了 getter 和 setter 以外的成员函数都不在实体里包括。

在 Spring Boot 框架下，这些代码被放入了控制器 Controller 或者服务层 Service 中了。

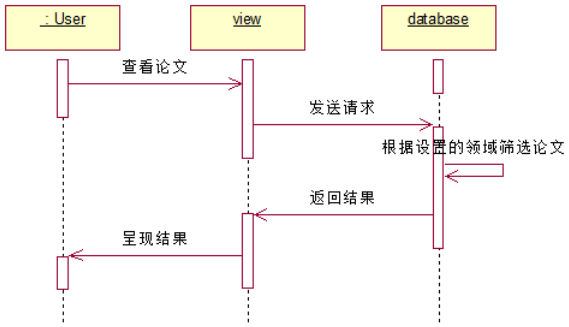
4.3. 时序图

4.3.1. 设置关注领域/关注实体



说明：用户面对视图，访问关注领域或者关注实体，视图向数据库（也是后台）发送请求，后台响应资源，视图将结果呈现给用户。而对于写请求，如保存选中领域，也是类似的时序。

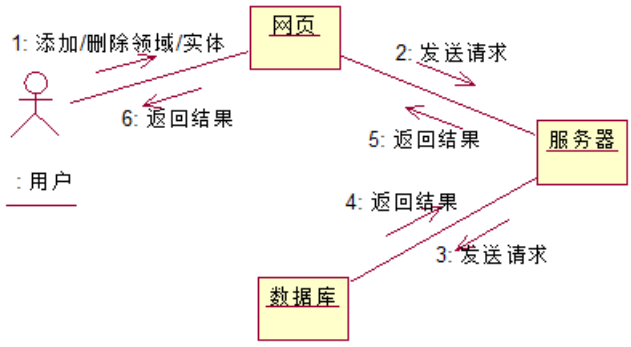
4.3.2. 查看论文



说明：用户面对视图，访问查看领域视图，视图向数据库（也是后台）发送请求，后台响应资源，视图将论文的查询结果呈现给用户。

4.4. 系统协作图

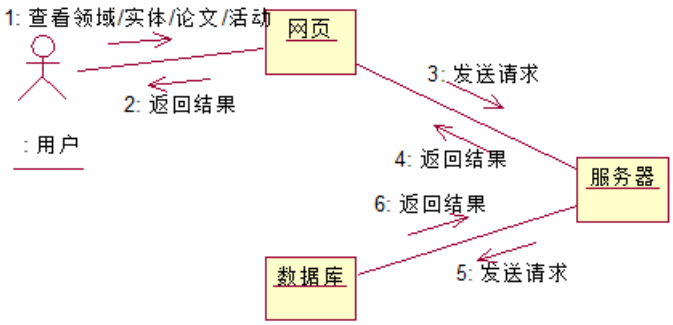
4.4.1. 设置领域/实体



说明：

用户在网页添加/删除领域/实体，网页、服务器、数据库依次发送请求，完成数据库查询后，结果层层返回给视图，视图再将结果呈现给用户。

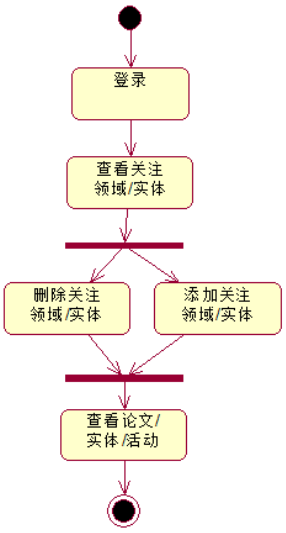
4.4.2. 查看论文/实体/活动/领域



说明：

用户在网页添加/删除领域/实体，网页、服务器、数据库依次发送请求，完成数据库查询后，结果层层返回给视图，视图再将结果呈现给用户。

4.5. 系统状态图

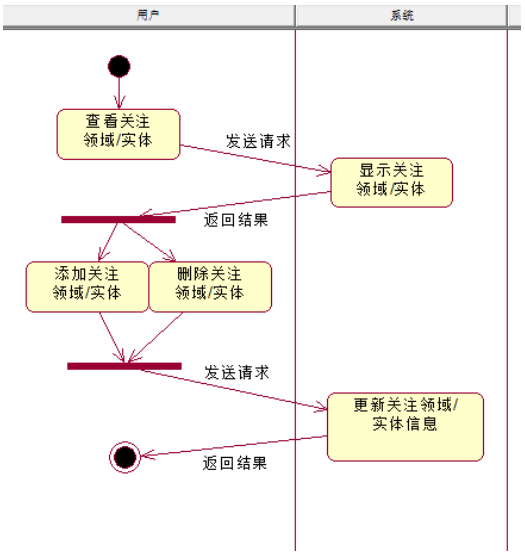


说明：

首先登录，在添加关注领域/实体后，可以查看论文/实体/学术活动。

4.6. 系统活动图

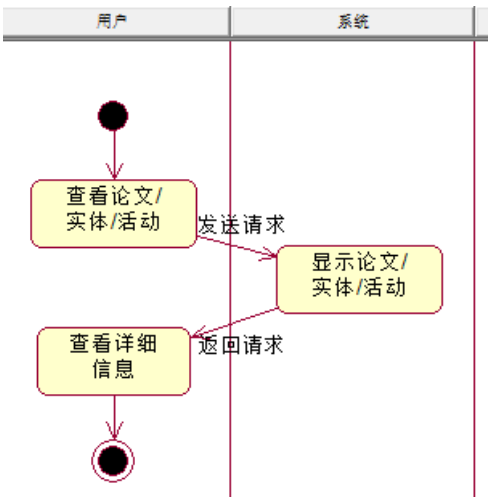
4.6.1. 查看并设置关注领域/实体



说明：

用户发出查看关注领域/实体的请求后，系统接受请求，返回关注领域/实体的资源；用户在看到相关资源后，可以作出相应的设计，如添加关注领域/实体，删除关注领域/实体，系统收到请求以后，更新自己的资源。

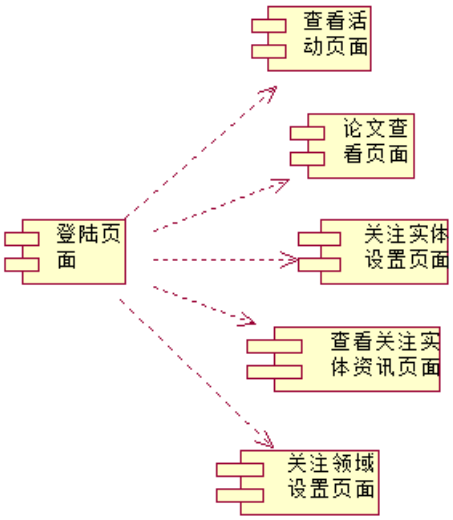
4.6.2. 查看论文/关注实体/学术活动



说明：

这一部分比较简单。用户查看论文/实体/活动，发送这样的请求，系统接受请求后，向用户返回结果，显示论文/实体/活动，用户可以查看详细信息。

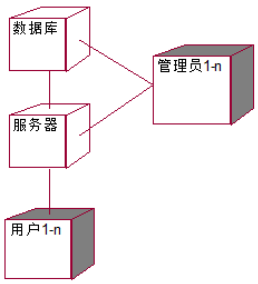
4.7. 系统组件图



说明：

一共有五个页面，查看活动页面、论文查看页面、关注实体设置页面、查看关注实体资讯页面、关注领域设置页面。这几个页面是平行的。

4.8. 系统配置图



系统配置图解释说明：

管理员可以访问数据库和服务器进行管理，而用户无法直接操作数据库。

4.9. 总体效果展示

左图是组件图和配置图，右图是其它所有的图。

5. 系统设计

5.1. 系统配置设计

5.1.1. 硬件配置设计

阿里云 ECS 服务器一台

vCPU	1
内存（GiB）	**2.0**
本地存储（GiB）	None
网络带宽能力（出/入）（Gbit/s）	**0.5**
网络收发包能力（出+入）（万 PPS）	5
支持 IPv6	No
多队列	1
弹性网卡（包括一块主网卡）	2
单块弹性网卡的私有 IP	2

5.1.2. 软件配置设计

操作系统：CentOS 7
数据库：MySQL 8.0
Java Web：Tomcat 9.0
JDK：1.8
核心框架：Spring Boot
数据库层：Spring Data JPA
安全框架：Shiro
数据库连接池：Druid
缓存：Ehcache

5.1.3. 网络配置设计

入方向设计：

授权策略	优先级	协议类型	端口范围	授权对象	描述
允许	1	自定义 TCP	目的:8080/8080	源:0.0.0.0/0	Tomcat 服务
允许	1	自定义 TCP	目的:3306/3306	源:0.0.0.0/0	MySQL 数据库远程访问端口
允许	1	自定义 TCP	目的:80/80	源:0.0.0.0/0	HTTP 协议端口

允许	110	全部 ICMP(IPv4)	目的:-1/-1	源:0.0.0.0/0	系统创建规则
允许	110	自定义 TCP	目的:22/22	源:0.0.0.0/0	系统创建规则
允许	110	自定义 TCP	目的:3389/3389	源:0.0.0.0/0	系统创建规则

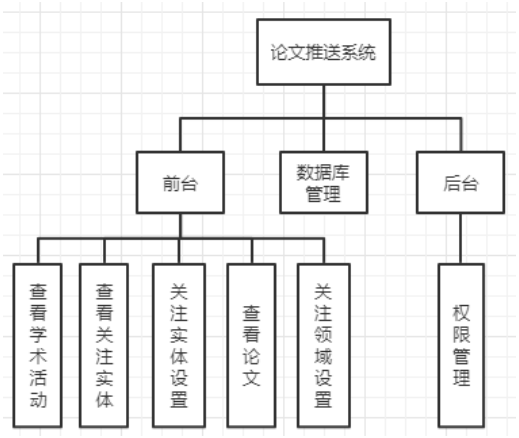
其中，授权对象的 0.0.0.0/0 指的是对于所有的 ip 地址；

端口范围 8080/8080 指的就是 8080 这个端口。

而为了安全性、流控和差控，全部是 TCP。

5.2. 系统结构设计

5.2.1. 系统功能结构设计

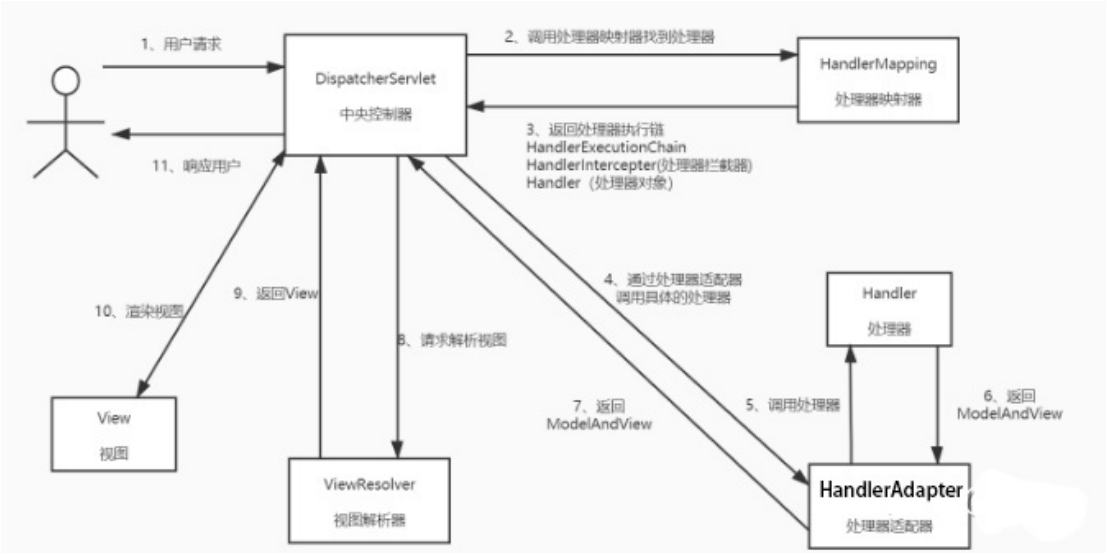


整个系统分为三大部分，分别是前台、后台和数据库管理。

前台主要实现了各类功能，数据库管理则是负责对数据库的访问，而后台是方便管理员进行管理。

5.2.2. 系统网络结构设计

这里是 Spring MVC 的网络结构设计。设计是如此设计，但实际上需要开发者动的东西其实往往是一个代码或者一个类。



说明：

用户发送请求至前端控制器 DispatcherServletDispatcherServlet

收到请求调用 HandlerMapping

处理器映射器处理器映射器根据请求 url 找到具体的处理器，生成处理器对象及处理器拦截器(如果有则生成)一并返回给 DispatcherServlet

DispatcherServlet 通过 HandlerAdapter 处理器适配器调用处理器

执行处理器(Controller，也叫后端控制器)

Controller 执行完成返回 ModelAndView

HandlerAdapter 将 controller 执行结果 ModelAndView 返回给 DispatcherServlet

DispatcherServlet 将 ModelAndView 传给 ViewReslover 视图解析器

ViewReslover 解析后返回具体 View

DispatcherServlet 对 View 进行渲染视图（即将模型数据填充至视图中）

DispatcherServlet 响应用户

而对应到程序员实际需要控制的部分：

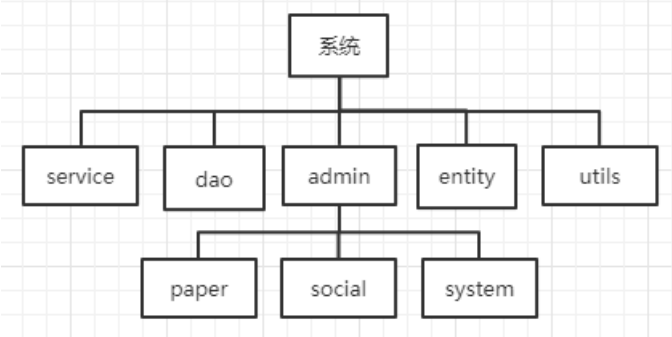
中央控制器 – 注释@Controller

视图解析器 – 方法 model.addView()

处理器适配器 – 注释@RequestMapping 标注的控制器成员函数

处理器映射器 – 注释@RequestMapping

5.3. 系统功能模块设计



5.3.1. Admin

整个系统的顶层与核心，管理一切调配。里面包含所有实体的控制器类。

5.3.2. Utils

工具包，包括一些字符串处理、校验等工具函数。一般工程上的实用性函数都会封装一层，而不会直接使用底层的工具函数，目的是解耦，方便替换。

5.3.3. Dao

数据访问层，专门负责对数据库的访问。数据访问层是一个代码类库，提供访问位于持久化容器中数据的功能，在分层设计中，所有从介质化读取数据或写入数据的工作都属于这一层的任务。

5.3.4. Entity

实体层，定义各类对象，也就是对应数据库里的一张表

5.3.5. Service

服务层，实现具体的业务。业务逻辑层中的模块包含了系统所需要的所有功能上的算法和计算过程，并与数据访问层和表现层交互，相当于中间类的作用，提供了另一个通用放任接口让调用者可以使用接口暴露的方法，而无需关注架构或底层发生的怎样的变化。

5.3.6. Paper

定义一系列论文推送相关具体对象，如 Paper，Subject 等。

5.3.7. Social

定义活动、关注对象等具体对象，如 SocialAccount 等。

5.3.8. System

定义用户、角色、权限等一系列对象，如 User，Role 等。

5.4. 编码设计

5.4.1. 数据编码

对于各类编码，新系统设计了一个 dict 表。

字典表的目的是对于一些有限制取值范围的字段进行存储。

比如，性别有男女之别，但如果对于每一个用户都设置一个字段表示为“男”或者“女”，有大量的语义重复，这时候把这些常用字段存储起来，能化简表格，减少表格之间的耦合度。

设置数据库表如下：

字段名	数据类型	允许为空	PK	字段说明
id	bigint	NO	自增	
create_by	bigint	YES		创建人
create_time	datetime	YES		创建时间/注册时间
modify_by	bigint	YES		最后更新人
modify_time	datetime	YES		最后更新时间
name	varchar(32)	YES		字典显示值
pid	bigint	YES		字典组 id
value	varchar(32)	YES		字典值

主要的字段是 id、name、pid、value 三个。

例如，对于性别，新系统就设置字典组 name=“性别”，value=0，其中包括 name=男，value=1；name=女，value=2；两个字典项。

id	name	pid	value
16	**状态**	**0**	**0**
17	启用	16	1
18	禁用	16	2
29	**性别**	**0**	**0**
30	男	29	1
31	女	29	2
35	**账号状态**	**0**	**0**
36	启用	35	1

37	冻结	35	2
38	已删除	35	3
53	**证件类型**	**0**	**0**
54	身份证	53	1
55	护照	53	2
68	**是否**	**0**	**0**
69	是	68	1
70	否	68	0
71	**消息类型**	**0**	**0**
72	短信	71	0
73	邮件	71	1

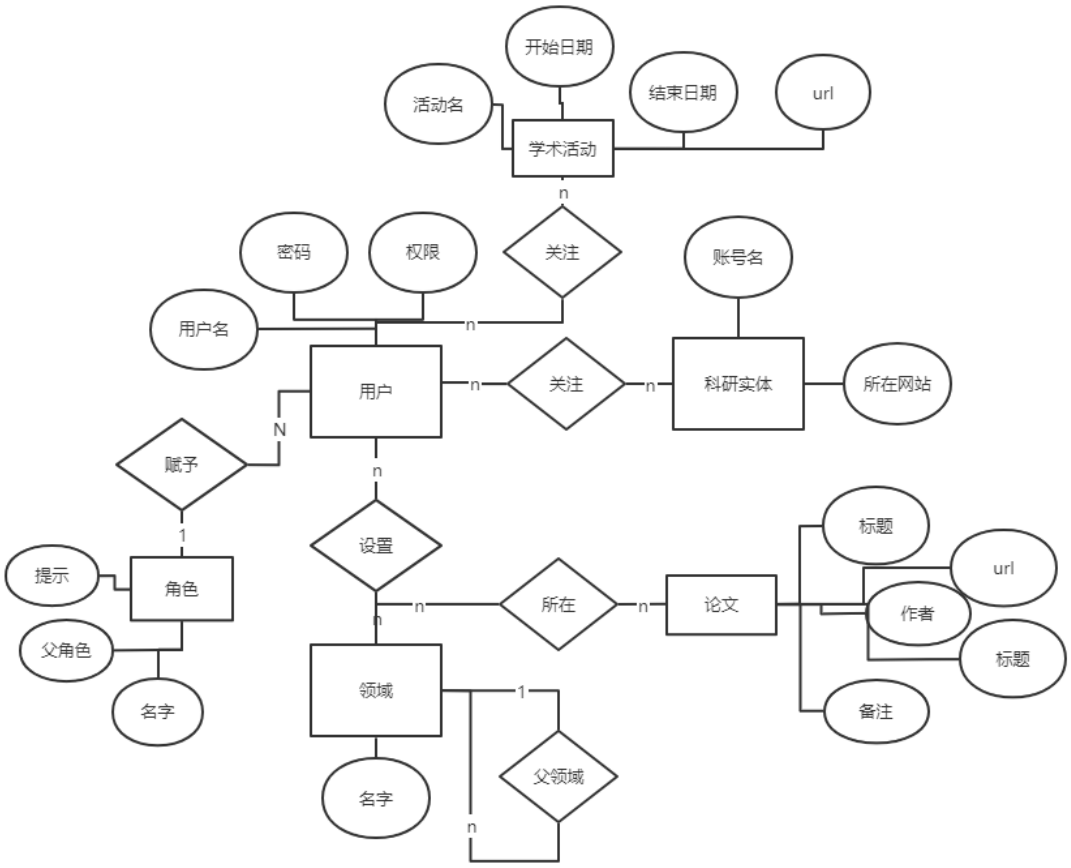
对于字典组，我设置了性别、账号状态、状态、证件类型、消息类型五个字典组。

注意如果它是字典组，那么它的 value 值默认为 0.

5.5. 数据库设计

5.5.1. 概念结构设计

说明本数据库将反映的现实世界中的实体、属性和它们之间的关系等的原始数据形式，包括各数据项、记录、系、文卷的标识符、定义、类型、度量单位和值域，建立本数据库的每一幅用户视图。



对一些重要关系进行说明：

一个用户可以关注多个领域，一个领域可以被多个用户关注，所以是 n-n；

一个科研实体可以被一个用户关注，一个用户可以关注多个科研实体，所以是 n-n；

一篇论文可以相关多个领域，一个领域可以有多篇论文，所以是 n-n。

一个角色只能赋予一个用户，所以是 1-n。

注：这里用户的属性太多，进行了省略。

5.5.2. 逻辑结构设计

说明把上述原始数据进行分解、合并后重新组织起来的数据库全局逻辑结构，包括所确定的关键字和属性、重新确定的记录结构和文卷结构、所建立的各个文卷之间的相互关系，形成本数据库的数据库管理员视图。

用户 User (id, username, password, roleid)

领域 Subject (id, name, pid)

科研实体 Account (id, name, website)

论文 Paper (id, title, author, comment, org, url)

学术活动 (id, name, start_time, end_time, url)

角色 (id, pid, name, tips)

用户_领域 (user_id, subject_id)

用户_实体 (user_id, account_id)

用户_学术活动 (user_id, activity_id)

5.5.3. 物理结构设计

建立系统程序员视图，包括：

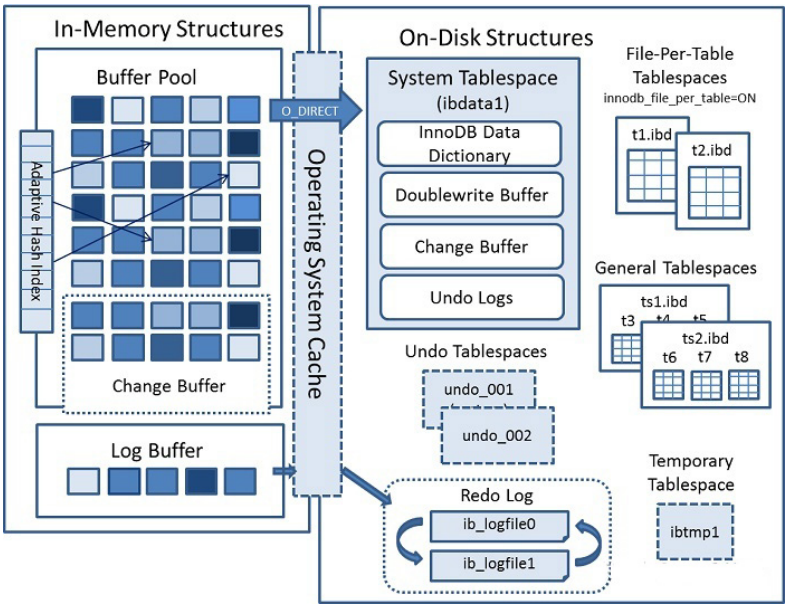
a. 数据在内存中的安排，包括对索引区、缓冲区的设计；

b. 所使用的外存设备及外存空间的组织，包括索引区、数据块的组织与划分；

c. 访问数据的方式方法。

新系统使用 MySQL 默认存储引擎 Innodb，故针对 InnoDB 进行说明。

内存中的安排



InnoDB 使用 malloc()操作在启动时为整个 Buffer Pool 分配缓存，缓冲池存储着常访问的表数据和索引，以页面为单位（一个页面可能包含多个行 ）组成一个链接列表，使用 LRU 的变体算法将缓冲池作为列表管理。

仅非主键索引的二级索引页不在 Buffer Pool 中时，会更改这部分缓存，将合并后读入 Buffer Pool 再刷到磁盘。

Log Buffer 保存要写入磁盘上的日志文件的数据。日志缓冲区大小由 innodb_log_buffer_size 变量定义 。默认大小为 16MB。Log Buffer 的内容会定期刷新到磁盘。较大的日志缓冲区使大型事务可以运行，而无需在事务提交之前将 redo log 数据写入磁盘。

外存设备及外存空间的组织

Innodb 只显式支持 B-Tree (从技术上来说是 B+Tree)索引，对于频繁访问的表，Innodb 会透明建立自适应 hash 索引，即在 B 树索引基础上建立 hash 索引，可以显著提高查找效率，对于客户端是透明的，不可控制的，隐式的。

访问数据的方式方法

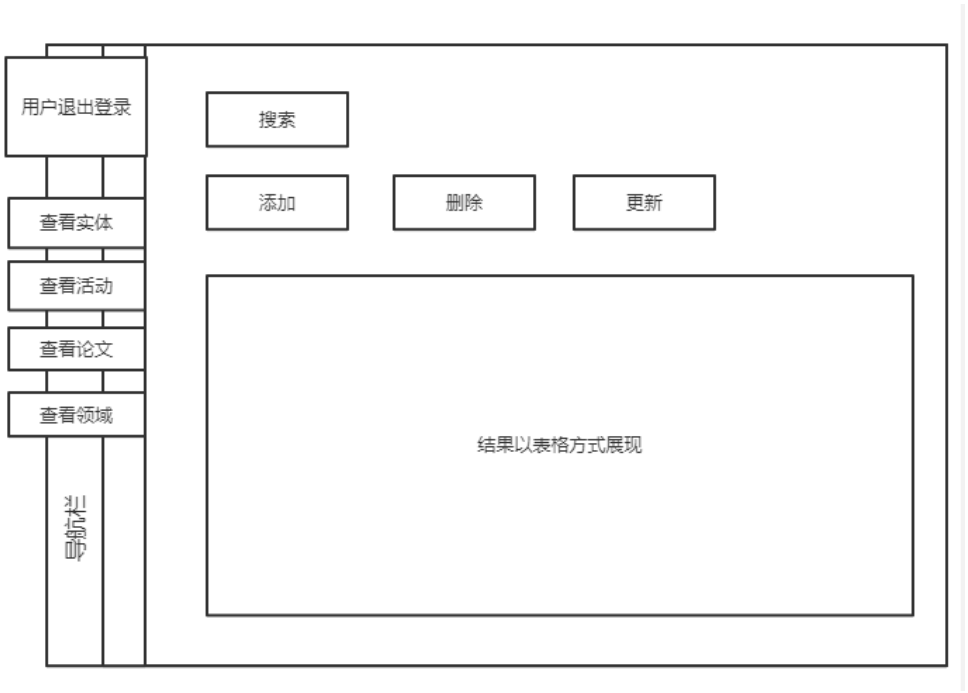
所有的数据项都有唯一识别的主键 id，在 id 字段上建立唯一索引。主键是一种特殊的唯一索引。一般情况下主键就是默认的聚簇索引。聚簇索引的顺序，就是数据在硬盘上的物理顺序。

5.6. 输入输出界面设计

网上有十分丰富的前端设计模板和框架，新系统使用 Bootstrap。Bootstrap 是用于开发响应式布局、移动设备优先的 WEB 项目，也是一个用于 HTML、CSS 和 JS 开发的开源工具包。利用 Bootstrap 提供的 Sass 变量和混合（mixins）、响应式栅格系统、可扩展的预制组件以及强大的 jQuery 插件，能够快速地开发出产品原型或构建整个 app。

在这里，我们使用一个国产 guns 框架，它有对 Bootstrap 组件进行更进一步的封装，已经有一个成熟的原型。

大体的输入输出界面如下：



左边的是用户导航栏，有新系统的各种选项；而右半边集中显示输入输出结果。

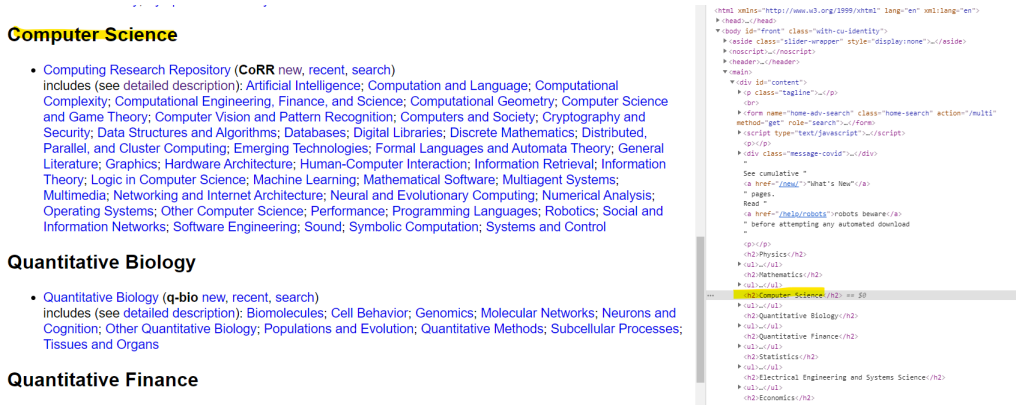
如果点击添加，会跳出页面：

5.7. 代码设计（主要程序代码片断）

5.7.1. arXiv 爬虫

新系统需要每日更新论文库，故需要将论文信息爬虫下来。arXiv 没有较强的反爬虫措施，也是目前更新最及时的计算机、航天、物理、数学等论文的网站，故先对 arXiv 进行爬虫。

首先爬的是学科分类：



Computer Science

- Computing Research Repository (**CoRR** new, recent, search)
  includes (see detailed description): Artificial Intelligence; Computation and Language; Computational Complexity; Computational Engineering, Finance, and Science; Computational Geometry; Computer Science and Game Theory; Computer Vision and Pattern Recognition; Computers and Society; Cryptography and Security; Data Structures and Algorithms; Databases; Digital Libraries; Discrete Mathematics; Distributed, Parallel, and Cluster Computing; Emerging Technologies; Formal Languages and Automata Theory; General Literature; Graphics; Hardware Architecture; Human-Computer Interaction; Information Retrieval; Information Theory; Logic in Computer Science; Machine Learning; Mathematical Software; Multiagent Systems; Multimedia; Networking and Internet Architecture; Neural and Evolutionary Computing; Numerical Analysis; Operating Systems; Other Computer Science; Performance; Programming Languages; Robotics; Social and Information Networks; Software Engineering; Sound; Symbolic Computation; Systems and Control

Quantitative Biology

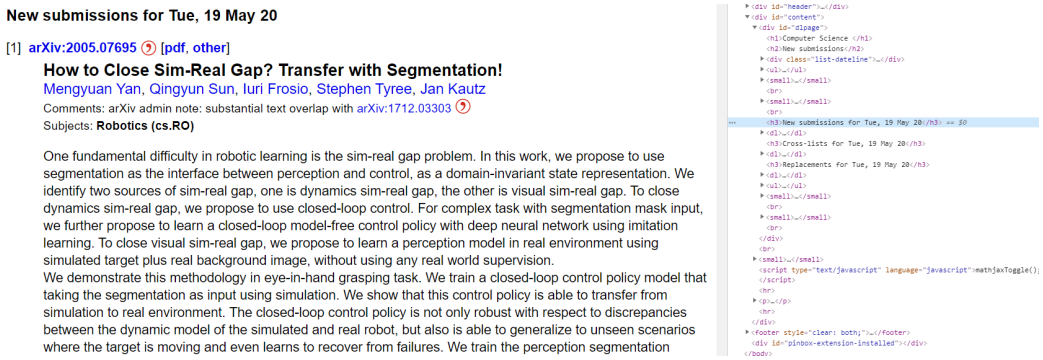
- Quantitative Biology (**q-bio** new, recent, search)
  includes (see detailed description): Biomolecules; Cell Behavior; Genomics; Molecular Networks; Neurons and Cognition; Other Quantitative Biology; Populations and Evolution; Quantitative Methods; Subcellular Processes; Tissues and Organs

Quantitative Finance

结果如下：

	A	B
1	subject	psubject
2	Artificial Intelligence	1
3	Computation and Language	1
4	Computational Complexity	1
5	Computational Engineering, Finance, and Science	1
6	Computational Geometry	1
7	Computer Science and Game Theory	1
8	Computer Vision and Pattern Recognition	1
9	Computers and Society	1
10	Cryptography and Security	1
11	Data Structures and Algorithms	1
12	Databases	1
13	Digital Libraries	1
14	Discrete Mathematics	1
15	Distributed, Parallel, and Cluster Computing	1
16	Emerging Technologies	1
17	Formal Languages and Automata Theory	1
18	General Literature	1
19	Graphics	1

而这个是每日论文：



New submissions for Tue, 19 May 20

[1] arXiv:2005.07695 [pdf, other]

How to Close Sim-Real Gap? Transfer with Segmentation!

Mengyuan Yan, Qingyun Sun, Iuri Frosio, Stephen Tyree, Jan Kautz

Comments: arXiv admin note: substantial text overlap with arXiv:1712.03303

Subjects: **Robotics (cs.RO)**

One fundamental difficulty in robotic learning is the sim-real gap problem. In this work, we propose to use segmentation as the interface between perception and control, as a domain-invariant state representation. We identify two sources of sim-real gap, one is dynamics sim-real gap, the other is visual sim-real gap. To close dynamics sim-real gap, we propose to use closed-loop control. For complex task with segmentation mask input, we further propose to learn a closed-loop model-free control policy with deep neural network using imitation learning. To close visual sim-real gap, we propose to learn a perception model in real environment using simulated target plus real background image, without using any real world supervision.
We demonstrate this methodology in eye-in-hand grasping task. We train a closed-loop control policy model that taking the segmentation as input using simulation. We show that this control policy is able to transfer from simulation to real environment. The closed-loop control policy is not only robust with respect to discrepancies between the dynamic model of the simulated and real robot, but also is able to generalize to unseen scenarios where the target is moving and even learns to recover from failures. We train the perception segmentation

主要爬取论文的代码如下：

```
    date = soup.find('h3')
list_ids = content.find_all('a', title = 'Abstract')
list_title = content.find_all('div', class_ = 'list-title mathjax')
list_authors = content.find_all('div', class_ = 'list-authors')
list_subjects = content.find_all('div', class_ = 'list-subjects')
list_subjects_split = []
list_authors_split = []
list_title_split = []
list_id = []
list_url_split = []
# modify url
for i in list_ids:

    i = i.text.split(':', maxsplit=1)[1]
    i.strip()
    list_id.append(int(i.split('.')[1]))
    list_url_split.append('arxiv.org/abs/'+i)

# modify title
for title in list_title:
    title = title.text.split(': ', maxsplit=1)[1]
    title = title.replace('\n', '')
    list_title_split.append(title)

# modify subject
for subjects in list_subjects:
    subjects = subjects.text.split(': ', maxsplit=1)[1]
    subjects = subjects.replace('\n\n', '')
    subjects = subjects.replace('\n', '')
    subject_split = subjects.split('; ')
    subject_split = [i.split(' (')[0] for i in subject_split]
    #print(subject_split)
```

```
    ids = pdSubjects[pdSubjects['name'].isin(subject_split)]['id']
    #print(ids)
    ids = list(ids)
    list_subjects_split.append(ids)

# modify author
for authors in list_authors:
    authors = authors.text.split(':', maxsplit=1)[1]
    authors = authors.replace('\n\n', '')
    authors = authors.replace('\n', '')
    authors = authors.replace(', ', ',')
    list_authors_split.append(authors)


items = []
for i, paper in enumerate(zip(list_id, list_url_split, list_title_split, list_authors_split, list_subjects_split)):
    items.append([paper[0], paper[1], paper[2], paper[3], paper[4]])
name = ['id', 'url', 'title', 'author', 'subject']
paper = pd.DataFrame(columns=name,data=items)
    paper.to_csv('./paper_'+time.strftime("%Y-%m-%d")+'_'+str(len(items))+'.csv')
```

结果如下：

```
    os.chdir(sys.path[0])
url = 'https://arxiv.org'
html = get_one_page(url)
soup = BeautifulSoup(html, features='html.parser')
content = soup.dl

psubjects = soup.find_all('h2')
pSubjectAbbr = 'cs'
subjects = soup.find_all('a', id=re.compile(r'cs.[A-Z]{2}') ) #, class_=re.compile(r'cs.[A-Z]{2}')
def saveCSV(subjects, psubjectId, psubjectName):
    items = []
    for i, s in enumerate(subjects):
        items.append([s.text, psubjectId])
    name = ['subject', 'psubject']
    paper = pd.DataFrame(columns=name,data=items,index=None)
    paper.to_csv('./'+psubjectName+'_'+str(len(items))+'.csv')
saveCSV(subjects, 1, 'Computer Science')
```

```
18.    pdSubjects = pd.read_csv('./subject.csv')
```

爬取了学科和论文后，还需要将学科统一编码，然后将论文所在的学科编码，储存进数据库。否则论文的学科是字符串，加大了存储的压力。这一步相当于联表的操作：

	A	B	C	D	E	F
1		id	url	title	author	subject
2	0	2380	arxiv.org/a	Bit-Interle	Sadjad Se	[27]
3	1	2384	arxiv.org/a	Compositi	Pawe脽 P.	[28]
4	2	2387	arxiv.org/a	SurvLIME-	Lev V. Utk	[29]
5	3	2389	arxiv.org/a	Jointly Spa	Wanqing .	[27]
6	4	2390	arxiv.org/a	Mechanisı	Akaki Mar	[11]
7	5	2392	arxiv.org/a	Deep Lagr	Matteo Ti	[29]
8	6	2426	arxiv.org/a	Communi	Zhishuai C	[19, 29]
9	7	2428	arxiv.org/a	Peak Pow	Mehiar Da	[33]
10	8	2431	arxiv.org/a	Automate	Ekaterina	[6, 7]
11	9	2433	arxiv.org/a	Stolen Prc	David Der	[29]
12	10	2434	arxiv.org/a	Nanotech	Randy Bry	[13, 20]
13	11	2435	arxiv.org/a	Effect of T	Deepak M	[29]
14	12	2436	arxiv.org/a	Data Augr	Hiroshi Sa	[12, 29]
15	13	2439	arxiv.org/a	Contextua	Brendan K	[7, 26, 29]
16	14	2440	arxiv.org/a	Evaluation	Azza E. A.	[33]
17	15	2441	arxiv.org/a	Physical L	Jos茅 Dav	[27]
18	16	2442	arxiv.org/a	Reliable ai	Jian Cao,N	[13]
19	17	2443	arxiv.org/a	A Dataset	Julio C. S.	[13, 41]
20	18	2449	arxiv.org/a	A Cloud-E	Peter Gor	[42, 45]
21	19	2450	arxiv.org/a	Iris segme	Abdullatif	[12]
22	20	2454	arxiv.org/a	Performar	Bassam Al	[33, 38]
23	21	2456	arxiv.org/a	Scalable a	Najmeddi	[14, 29, 45]
24	22	2457	arxiv.org/a	Dynamic S	Bechir Ha	[33]
25	23	2459	arxiv.org/a	Deep Rein	Ming Tan	[33]

将上面的表提取出 paper_id 和 subject_id，最终结果如下：

	A	B	C	D
1	paper_id	subject_id		
2	2380	27		
3	2384	28		
4	2387	29		
5	2389	27		
6	2390	11		
7	2392	29		
8	2426	19		
9	2426	29		
10	2428	33		
11	2431	6		
12	2431	7		
13	2433	29		
14	2434	13		
15	2434	20		
16	2435	29		
17	2436	12		
18	2436	29		
19	2439	7		
20	2439	26		
21	2439	29		
22	2440	33		
23	2441	27		
24	2442	13		
25	2443	13		
26	2443	41		
27	2449	42		
28	2449	45		
29	2450	12		
30	2454	33		

paper_subject_2020-05-07_199

此表是 paper 和 subject 的中间表。

5.7.2. 爬取学术会议

ALLCOMFERENCES 是集合了几乎所有学科学术会议的网站，同样也需要爬下来更新学术活动表。主要代码如下：

```
    os.chdir(sys.path[0])
url = 'http://www.allconferences.com/search/index/Category__parent_id:12290/Conference__start_date__from:05-01-2020/Conference__start_date__to:11-01-2020/showLastConference:1/'
html = get_one_page(url)
soup = BeautifulSoup(html, features='html.parser')
content = soup.dl

conferences = soup.find_all('div', id=re.compile(r'conference_\d+'))

def timeTransfer(origin):
    tmp = time.strptime(origin, "%b %d, %Y")
    return time.strftime("%Y-%m-%d %H:%M:%S", tmp)

activities = []
for c in conferences:
    beginTime = c.find('span', class_='begin_txt').text.split('\n',maxsplit=2)[1]
    beginTime = timeTransfer(beginTime)
    endTime = c.find('span', class_='end_txt').text.split('\r\n',maxsplit=1)[1].strip()
    endTime = timeTransfer(endTime)
    url = c.find('h2').find('a')
    name = url.text
    if name == None:
        continue
    url = url.get('href')
    activities.append([name, url, beginTime, endTime])
activity = pd.DataFrame(columns=['name', 'url', 'beginTime', 'endTime'],data=activities)
    activity.to_csv('./activity_'+time.strftime("%Y-%m-%d")+'_'+str(len(activities))+'.csv',index=None)
```

过程中注意时间格式的转换。

结果如下：

	A	B	C	D
1	name	url	beginTime	endTime
2	IEEE--2020 The 5th International Conference on Big Data Analytics (ICBDA 2020)--Ei,Scopus	http://ww	2020/5/8 0:00	2020/5/11 0:00
3	ACM--2020 the 4th Int. Conference on Innovation in Artificial Intelligence (ICIAI 2020)--Ei,Scopus	http://ww	2020/5/8 0:00	2020/5/11 0:00
4	2020 The 7th International Conference on Digital Manufacturing and Automation (ICDMA 2020)-EI,Scopus	http://ww	2020/5/10 0:00	2020/5/12 0:00
5	2020 International Conference on Wireless Communication and Sensor Networks (icWCSN 2020)--Ei,Scopus	http://ww	2020/5/13 0:00	2020/5/15 0:00
6	2020 World Conference on Computing and Communication Technologies (WCCCT 2020)--Ei,Scopus	http://ww	2020/5/13 0:00	2020/5/15 0:00
7	2020 IEEE 5th International Conference on Computer and Communication Systems (ICCCS 2020)--Ei,Scopus	http://ww	2020/5/15 0:00	2020/5/17 0:00
8	2020 The Int. Conference on Information Technology and Internet of Things (ITIOT 2020)--Ei,Scopus	http://ww	2020/5/15 0:00	2020/5/17 0:00
9	SPIE--2020 The 12th International Conference on Digital Image Processing (ICDIP 2020)--Ei,Scopus	http://ww	2020/5/19 0:00	2020/5/22 0:00
10	2020 3rd International Conference on Robotics and Intelligent Technology (ICRIT 2020)--JA, Scopus	http://ww	2020/5/20 0:00	2020/5/22 0:00
11	ACM--2020 5th Int. Conf. on Multimedia Systems and Signal Processing (ICMSSP 2020)--Ei,Scopus	http://ww	2020/5/28 0:00	2020/5/30 0:00
12	ACM--2020 5th International Conference on Big Data and Computing (ICBDC 2020)--Ei,Scopus	http://ww	2020/5/28 0:00	2020/5/30 0:00
13	IEEE--2020 3rd Int. Conference on Artificial Intelligence and Big Data (ICAIBD 2020)--Ei,Scopus	http://ww	2020/5/28 0:00	2020/5/31 0:00
14	SPIE--2020 the 5th International Workshop on Pattern Recognition (IWPR 2020)--Ei,Scopus	http://ww	2020/6/5 0:00	2020/6/7 0:00
15	2020 The 5th Int. Conference on Knowledge Engineering and Applications (ICKEA 2020)--Ei,Scopus	http://ww	2020/6/5 0:00	2020/6/8 0:00
16	2020 3rd International Conference on Communication and Network Protocol (ICCNP 2020)--JA,SCOPUS	http://ww	2020/6/10 0:00	2020/6/12 0:00
17	2020 The Int. Conference on Computer Science, Engineering and Education (CSEE 2020)--EI,Scopus	http://ww	2020/6/10 0:00	2020/6/12 0:00
18	2020 The 3rd International Conference on Data Storage and Data Engineering (DSDE 2020)--Scopus,Ei	http://ww	2020/6/10 0:00	2020/6/12 0:00
19	2020 IEEE 12th Int. Conference on Communication Software and Networks (ICCSN 2020)--Ei,Scopus	http://ww	2020/6/12 0:00	2020/6/15 0:00
20	IEEE--2020 8th International Conference on Wireless and Optical Communications(ICWOC 2020)-Ei,Scopus	http://ww	2020/6/12 0:00	2020/6/15 0:00
21	2020 4th IEEE International Conference on Robotics and Automation Sciences (ICRAS 2020)--Ei,Scopus	http://ww	2020/6/14 0:00	2020/6/16 0:00
22				

然后可以很方便地通过 navicat 导入数据库。Mysql 也有导入.csv 文件的指令。

5.7.3. 以 SocialAccount 显示为例，说明 Controller 编写

Controller 是 Spring Boot 中的核心类。

```
1.    @Controller
2.  @Slf4j
3.  @RequestMapping("/social/account")
4.  public class SocialAccountController extends BaseController {
5.      @Autowired
6.      private SocialAccountService saService;
7.      @Autowired
8.      private UserService userService;
9.
10.     private static String PREFIX = "/social/account/";
11.
12.
13.     @RequestMapping("")
14.     public String index() {
15.         return PREFIX + "account.html";
16.     }
17.
18.     // 展示社交账号
19.     @RequestMapping("/list")
20.     @ResponseBody
21.     public Object list(@RequestParam(required = false) String username) {
22.
23.         Page <SocialAccount> page = new PageFactory<SocialAccount>().default
    Page();
24.         Long userId = ShiroKit.getUser().getId();
25.         User user = userService.get(userId);
26.         List <SocialAccount> rsl;
27.         if(StringUtils.isNotEmpty(username)){
```

```
28.             rsl = saService.queryAllByUserIdAndLikeUsername(userId, username
    );
29.         }
30.         else {
31.             rsl = new ArrayList<>(user.getSocialAccounts()); //saService.que
    ryAllByUserId(userId);
32.         }
33.
34.         page.setRecords(rsl);
35.         page.setTotal(Integer.valueOf(rsl.size()+""));
36.
37.         return packForBT(page);
38.     }
39.
40.     //跳转到添加参数
41.     @RequestMapping("/account_add")
42.     public String add() {
43.         return PREFIX + "account_add.html";
44.     }
45.
46.     // 新增参数
47.     @RequestMapping(value = "/add")
48.     @ResponseBody
49.     public Object add(SocialAccount sa) {
50.         saService.update(sa);
51.
52.         Long id = ShiroKit.getUser().getId();
53.         log.info("User: " + id);
54.         User user = userService.get(id);
55.         Set<SocialAccount> userSa = user.getSocialAccounts();
56.         userSa.add(sa);
57.         userService.update(user);
58.
59.         return SUCCESS_TIP;
60.     }
61.
62.
63.     // 删除
64.     @RequestMapping(value = "/delete")
65.     @ResponseBody
66.     public Object delete(@RequestParam Long socialAccountId) {
67.         saService.delete(socialAccountId);
68.
69.         Long userId = ShiroKit.getUser().getId();
```

```
70.         log.info("User: " + userId);
71.         User user = userService.get(userId);
72.         Set<SocialAccount> userSa = user.getSocialAccounts();
73.         for (SocialAccount e: userSa) {
74.             if (e.getId() == socialAccountId) {
75.                 userSa.remove(e);
76.                 break;
77.             }
78.         }
79.         user.setSocialAccounts(userSa);
80.
81.         return SUCCESS_TIP;
82.     }
83.
84.    // 跳转到修改
85.     @RequestMapping("/account_update/{saId}")
86.     public String update(@PathVariable Long saId, Model model) {
87.         SocialAccount sa = saService.get(saId);
88.         model.addAttribute("item", sa);
89.         return PREFIX + "account_edit.html";
90.     }
91.
92.     @RequestMapping(value = "/update")
93.     @ResponseBody
94.     public Object update(SocialAccount sa) {
95.         saService.update(sa);
96.         return SUCCESS_TIP;
97.     }
98.    }
```

说明：

@Controller：将这个类声明为 Controller；

@RequestMapping("/social/account")：如果收到来自 view 的为 social/account 的访问请求，则激活这个控制器；

Autowired：

@Autowired

private SocialAccountService saService;

@Autowired

private UserService userService;

这里利用了 Spring Boot 的特性。上面的控制器中没有这两个成员的 new 代码，但

Spring 会帮我们自动连接，也就是依赖注入。这种方法大大降低了程序的耦合性，是伟大的发明创造。

查

```
// 展示社交账号
    @RequestMapping("/list")
    @ResponseBody
    public Object list(@RequestParam(required = false) String username) {

        Page <SocialAccount> page = new PageFactory<SocialAccount>().defaultPage();
        Long userId = ShiroKit.getUser().getId();
        User user = userService.get(userId);
        List <SocialAccount> rsl;
        if(StringUtils.isNotEmpty(username)){
            rsl = saService.queryAllByUserIdAndLikeUsername(userId, username);
        }
        else {
            rsl = new ArrayList<>(user.getSocialAccounts()); //saService.queryAllByUserId(userId);
        }

        page.setRecords(rsl);
        page.setTotal(Integer.valueOf(rsl.size()+""));

        return packForBT(page);
    }
```

RequestMapping(“/list”)表示如果收到来自 view 的为 social/account/list 的访问请求，则激活这个函数来响应。

函数参数里的@RequestParam(required = false) String username 是指前端传过来的数据里有搜索功能，这个函数参数则是获取搜索框的值。

Page <SocialAccount> page：声明一个页面变量，可以存放数据条目，是一种封装；

ShiroKit 是权限控制，可以通过它拿到用户 id。

再通过 userService 查询数据库，获取用户对象。

如果 username 有值，则说明需要对 username 进行模糊匹配。

如果 username 无值，则可以直接通过 User 内的 socialAccounts 取值。这是 Spring Data JPA 的好处，其实它相当于底层对数据库的联表查询。具体设置如下：

多对多映射实现

在 SSM 框架时代，多对多映射可以写 xml 也可以注释，而 Spring Boot 只用注释。如下是 User 结构的一部分：

```
    @ManyToMany(targetEntity = SocialAccount.class)
@JoinTable(name = "user_social_account", joinColumns = @JoinColumn(name = "user_id"),
inverseJoinColumns = @JoinColumn(name = "social_account_id"))
    private Set <SocialAccount> socialAccounts = new HashSet<>();
```

ManyToMany 声明了多对多映射，也就是 SocialAccount 和 User 的多对多；

JoinTable 的 name 参数声明了数据库里的中间表；

而最精彩的就是 socialAccounts 的设计。这里把软件和数据库访问统一了。也就是如果你访问这里的 socialAccount，对应于底层就相当于访问 user、social_account 和它们的中间表的联合查询。

增

```
// 新增参数
    @RequestMapping(value = "/add")
    @ResponseBody
    public Object add(SocialAccount sa) {
        saService.update(sa);

        Long id = ShiroKit.getUser().getId();
        log.info("User: " + id);
        User user = userService.get(id);
        Set<SocialAccount> userSa = user.getSocialAccounts();
        userSa.add(sa);
        userService.update(user);

        return SUCCESS_TIP;
    }

```

对于来自 social/account/add 的请求，则调用这个函数。由于每个用户的关注实体都不一样，于是要先获得用户。

函数参数是来自前端的传输。

删

```
// 删除
    @RequestMapping(value = "/delete")
```

```
    @ResponseBody
    public Object delete(@RequestParam Long socialAccountId) {
        saService.delete(socialAccountId);

        Long userId = ShiroKit.getUser().getId();
        log.info("User: " + userId);
        User user = userService.get(userId);
        Set<SocialAccount> userSa = user.getSocialAccounts();
        for (SocialAccount e: userSa) {
            if (e.getId() == socialAccountId) {
                userSa.remove(e);
                break;
            }
        }
        user.setSocialAccounts(userSa);

        return SUCCESS_TIP;
    }
```

可以看到，这里并没有访问数据库的语句，直接通过软件代码即可完成删除工作。

改

```
@RequestMapping("/account_update/{saId}")
    public String update(@PathVariable Long saId, Model model) {
        SocialAccount sa = saService.get(saId);
        model.addAttribute("item", sa);
        return PREFIX + "account_edit.html";
    }

    @RequestMapping(value = "/update")
    @ResponseBody
    public Object update(SocialAccount sa) {
        saService.update(sa);
        return SUCCESS_TIP;
    }
    }
```

如果点击更改按钮，那么就会返回给用户 html 页面，将用户填写的集合成 SocialAccount 对象，然后更新即可。

5.7.4. 以 paper 显示为例，数据库访问

虽然 Spring Data JPA 为我们提供了很多很方便的工具让我们少写 mysql 代码，但它有其局限性，就是不够灵活。比如对于 paper 的筛选需要通过 user、subject、paper 以及它

们的中间表，使用软代码效率很低。故需要老老实实写 MySQL。

前文提到，Repository 类是访问数据库的底层：

```
public interface PaperRepository extends BaseRepository<Paper, Long> {
    // 前面必须把 paper 的所有属性写出来，否则返回 List<Paper>的时候会拿错误的字段去生成 Paper
    // 造成出现重复论文的错误
    @Query(nativeQuery = true, value =
    "SELECT paper.title, paper.author, paper.id, paper.create_by, paper.create_time, paper.modify_time, paper.modify_by, paper.title, paper.url, paper.org, paper.comment"
    +" FROM ( `subject` INNER JOIN user_subject ON (user_subject.user_id=1 AND subject.id=user_subject.subject_id) ) INNER JOIN paper_subject ON paper_subject.subject_id=user_subject.subject_id INNER JOIN paper ON paper.id=paper_subject.paper_id"
    )
    List<Paper> findByUserId(Long userId);
}
```

将@Query 的 MySQL 语句放在函数上面，就是代表调用这个语句的时候执行这个 MySQL 语句，并返回结果到 Paper 的列表。

其中的 MySQL 代码连续使用了三个 Inner join：

SELECT paper.title, paper.author, paper.id, paper.create_by, paper.create_time, paper.modify_time, paper.modify_by, paper.title, paper.url, paper.org, paper.comment

FROM

(`subject` INNER JOIN user_subject ON (user_subject.user_id=1 AND subject.id=user_subject.subject_id))

INNER JOIN paper_subject ON paper_subject.subject_id=user_subject.subject_id

INNER JOIN paper ON paper.id=paper_subject.paper_id

前文提到，Service 负责调用 Repository：

```
@Service
public class PaperService extends BaseService<Paper, Long, PaperRepository>{

    @Autowired
    private PaperRepository paperRepo;

    public List<Paper> findByUserId(Long userId){
        return paperRepo.findByUserId(userId);
    }
    ...
}
```

@Service 表明了这是一个 Service；

@Autowired 是 paperRepository 的依赖注入。

在 findByUserId 内部调用 Repository 的 findByUserId 即可。

而在控制器里，调用 Service：

```
1.    @Controller
2.  @RequestMapping("/paper/paper")
3.  public class PaperController extends BaseController{
4.      @Autowired
5.      private PaperService paperService;
6.      @Autowired
7.      private UserService userService;
8.
9.      @RequestMapping(value = "/list")
10.     @ResponseBody
11.     public Object list(@RequestParam(required = false) String title) {
12.         Page <Paper> page = new PageFactory<Paper>().defaultPage();
13.         Long userId = ShiroKit.getUser().getId();
14.         List<Paper> rsl = paperService.findByUserId(userId);
15.         page.setRecords(rsl);
16.         page.setTotal(Integer.valueOf(rsl.size()+""));
17.         return packForBT(page);
18.     }
19.   }
```

可以看到标红的代码调用了 Service 中的查询函数。

5.7.5. 总结

SocialAccount 这个例子基本涵盖了增删查改，前端后端。其它功能基本也是上面的实现套路，在过程中可以体会 Spring Boot 相对于 SSM 框架的优越性。

6. 系统测试

6.1. 系统测试环境

由于 Java 一次编译处处运行的机制，新系统在本地完成测试、编译以及打包，再上传到服务器。

6.1.1. 测试的硬件环境

6.1.2. 测试的软件环境

操作系统：Windows 10
数据库：MySQL 8.0
数据库可视化：Navicat 12
Java Web：Tomcat 9.0
JDK：1.8
核心框架：Spring Boot
数据库层：Spring Data JPA
安全框架：Shiro
数据库连接池：Druid
缓存：Ehcache
浏览器：Chrome

6.1.3. 测试的网络环境

本地测试，无需联网

6.2. 系统测试内容

测试以下功能是否正常：

6.2.1. 登录，个人信息查看，退出

如果输入错误，是否有错误提示；
能否显示个人信息；
能否退出；

6.2.2. 设置领域

如果是已经设置的领域，需要标明；
如果是未设置的领域，则不需要标明；

6.2.3. 查看论文

在改变领域设置前后，论文的显示是否有变化；

6.2.4. 查看学术活动

学术活动能否搜索；

6.2.5. 查看关注实体

能否修改关注实体；
能否删除关注实体；
能否添加关注实体；

6.2.6. 管理用户

能否新增用户；
能否给用户赋予角色；
能否删除用户；

6.3. 系统测试方法

对于自己编写的代码，使用白盒测试，由于 Spring 的高度解耦，基本没有程序逻辑功能上的分支。

对于框架，使用黑盒测试，若出现问题，当框架外的部分测试认为无误后，再测试框架内部。

对于 Java 包，正确性已经经过社区考验，不需要测试。

6.3.1. 登录

如果密码输入错误，提示错误。

使用正确的密码登录，一共有两个账号供演示：

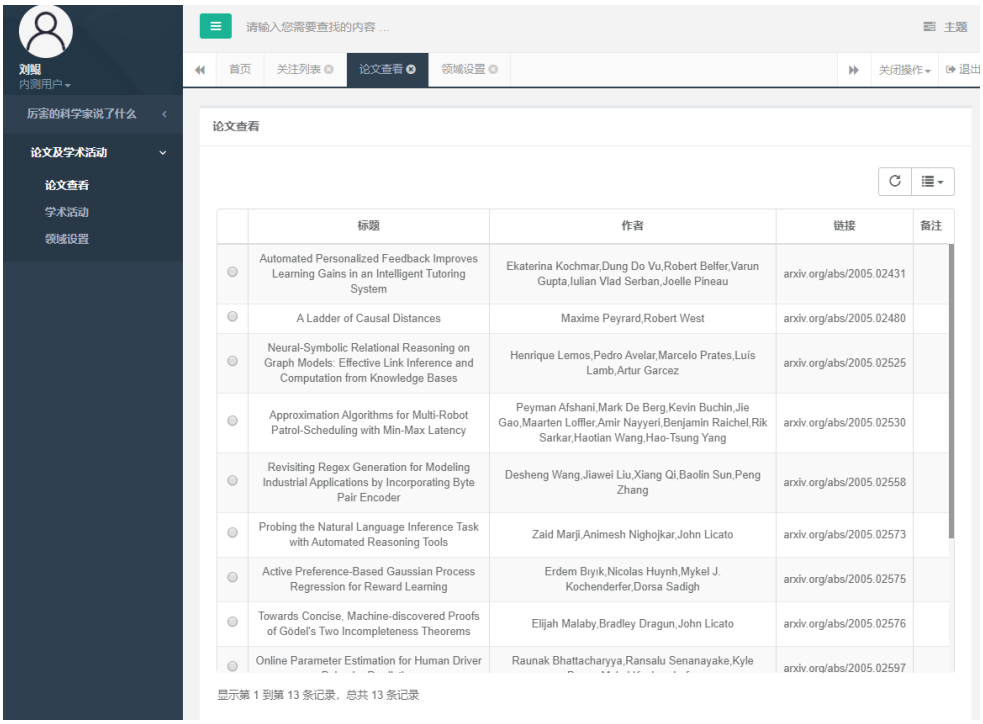
内测用户：账号：liukun，密码：111111

管理员用户：账号：developer，密码：111111

登录后界面如下：

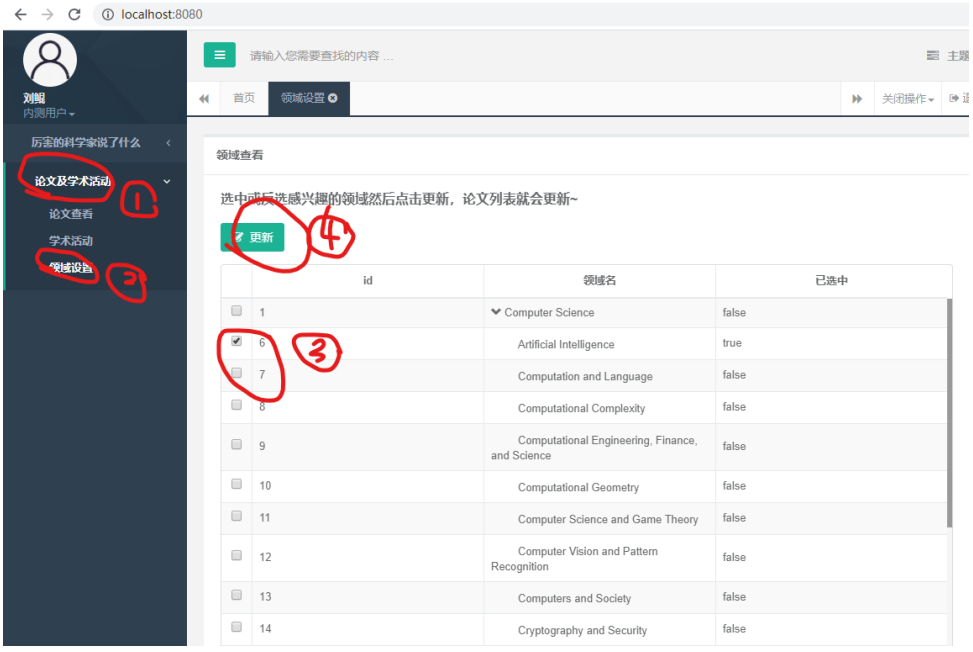
6.3.2. 查看论文

点击“论文及学术活动”->“论文查看”，可以看到有 13 篇论文。

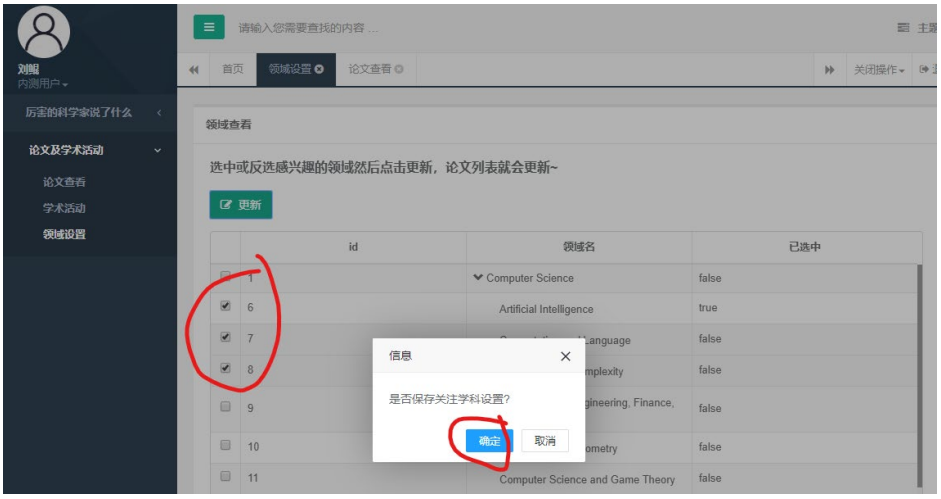


6.3.3. 设置领域

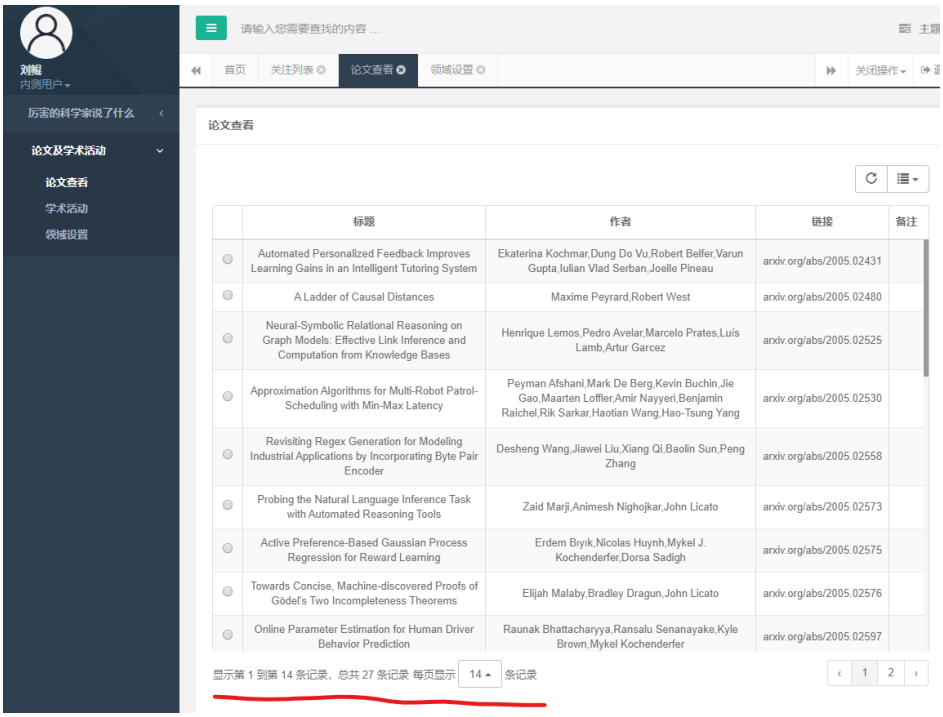
点击“论文及学术活动”->“领域设置”，点击 7、8：



点击更新：

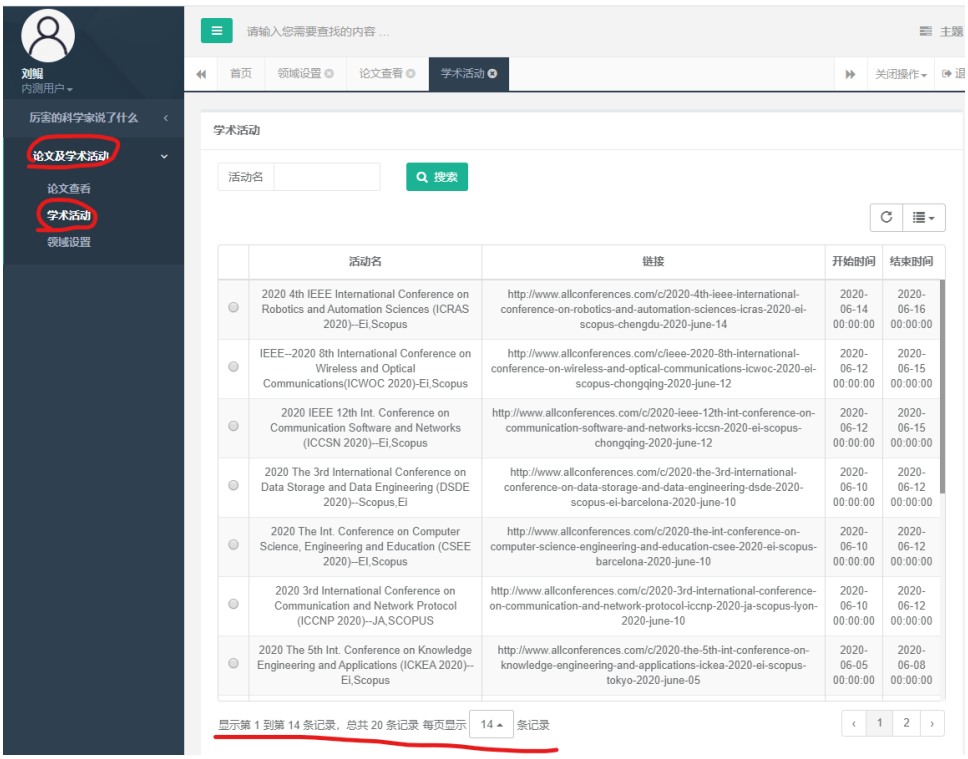


保存后，再次点击“论文查看”：



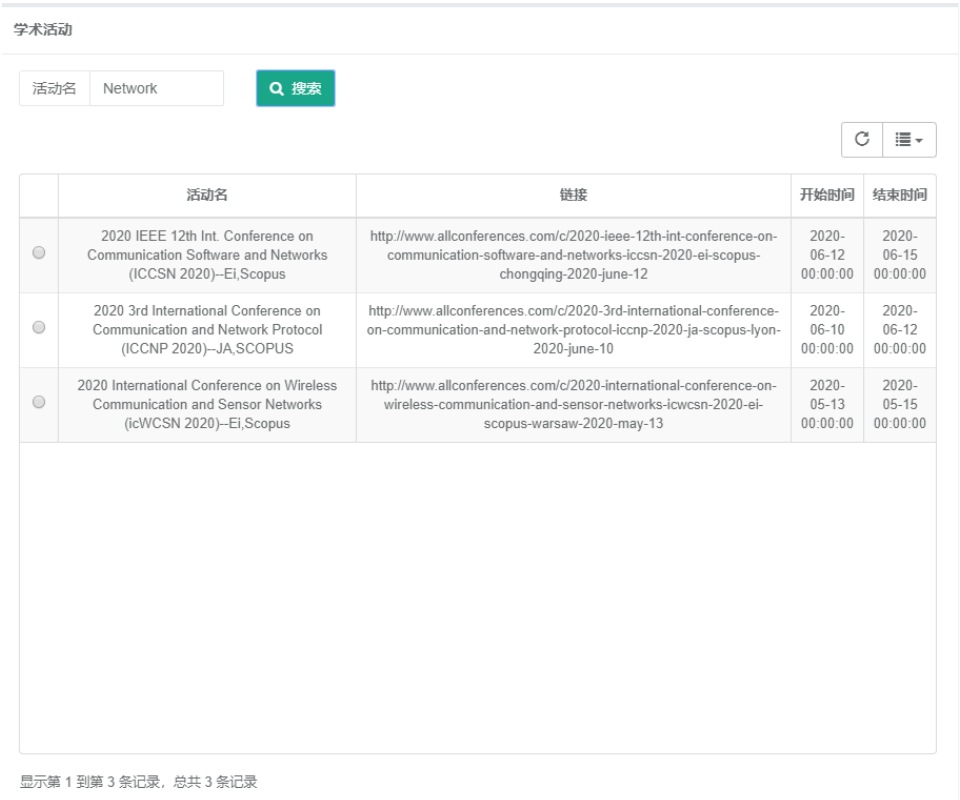
可以看到，论文数量增多了。说明这部分通过测试。

6.3.4. 查看学术活动



导航栏点击论文及学术活动->学术活动，可以看到有 14 页的记录。

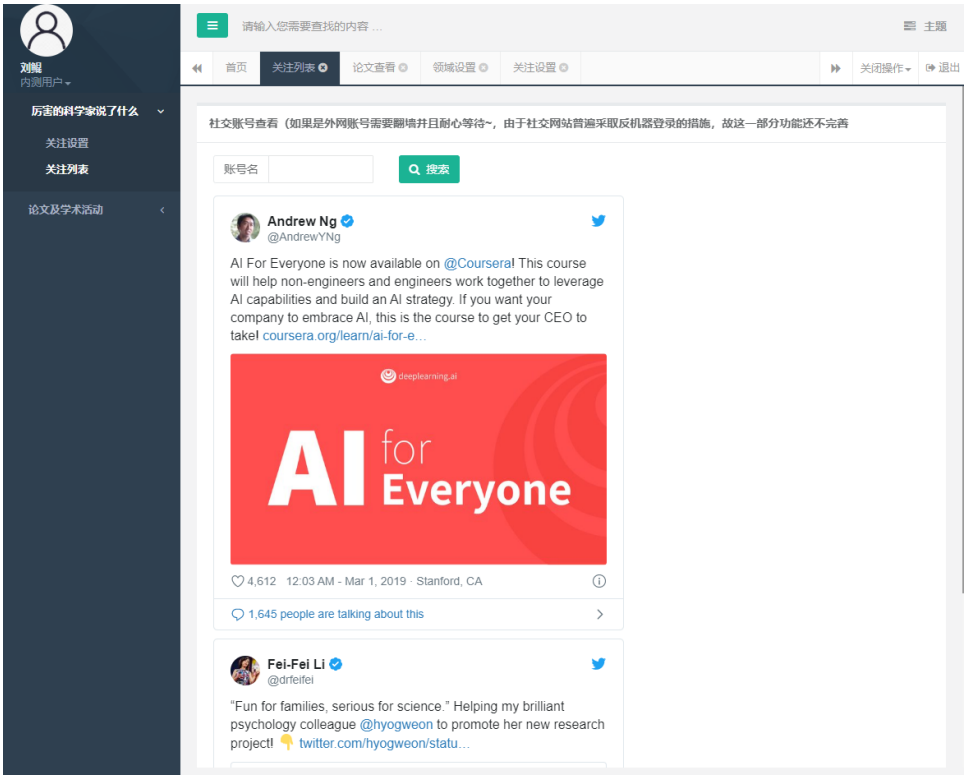
在搜索框内输入 Network，可以查找 Network 相关的学术活动：



6.3.5. 查看关注实体

点击“厉害的科学家说了什么”->“关注设置”，就可以看到吴恩达和李飞飞的账

号。

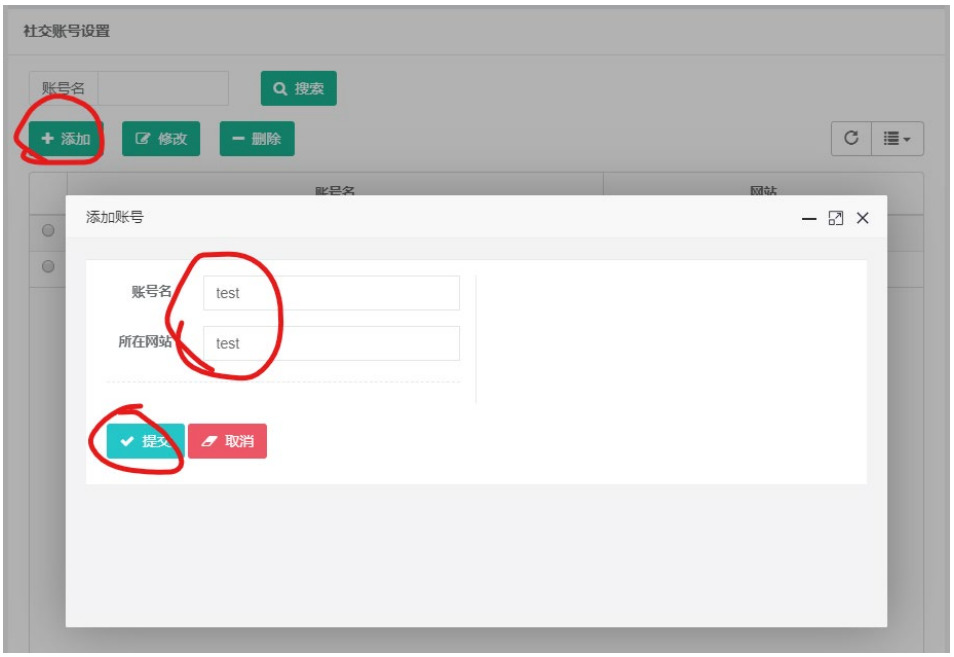


如果是外网账号需要翻墙并且耐心等待，由于社交网站普遍采取反机器登录的措施，故这一部分功能还不完善。

6.3.6. 设置关注实体

点击“厉害的科学家说了什么”，点击“关注设置”，可以看到当前用户关注的社交账号。

点击添加，即可设置账号名和所在网站：



点击提交后，结果如下：

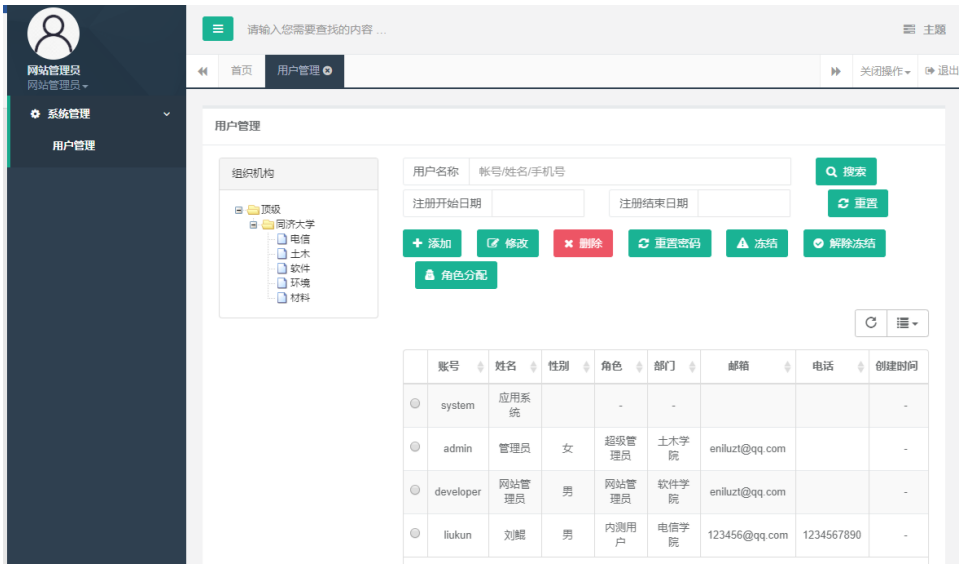
可以看到，成功新增了账号。

6.3.7. 管理用户

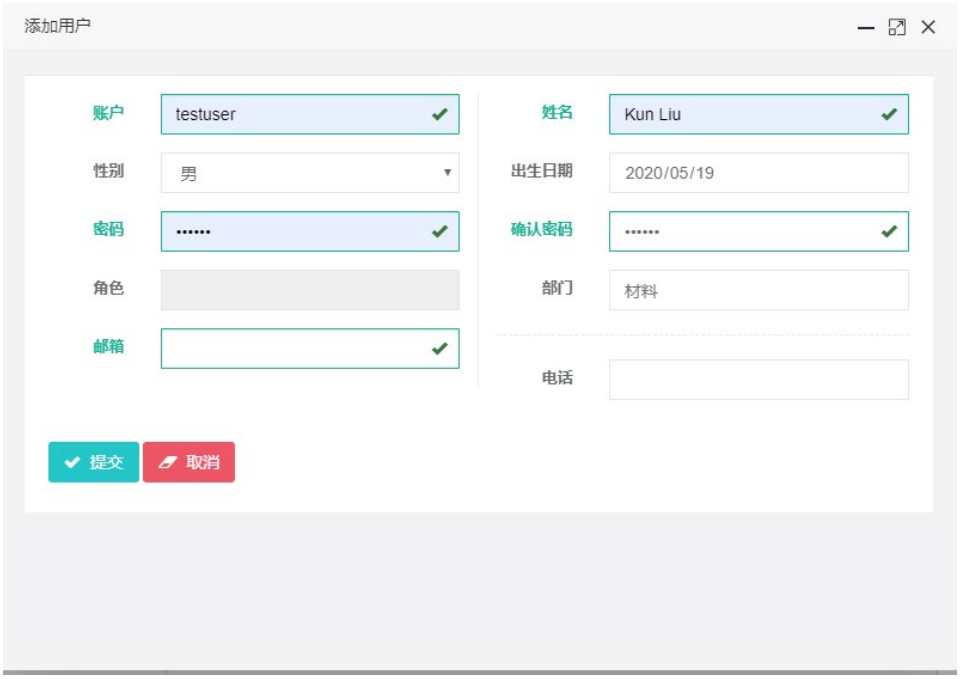
重新登录一个管理员账号：developer，密码：111111：

可以看到系统管理选项。

点击“系统管理”->“用户管理”，可以看到：



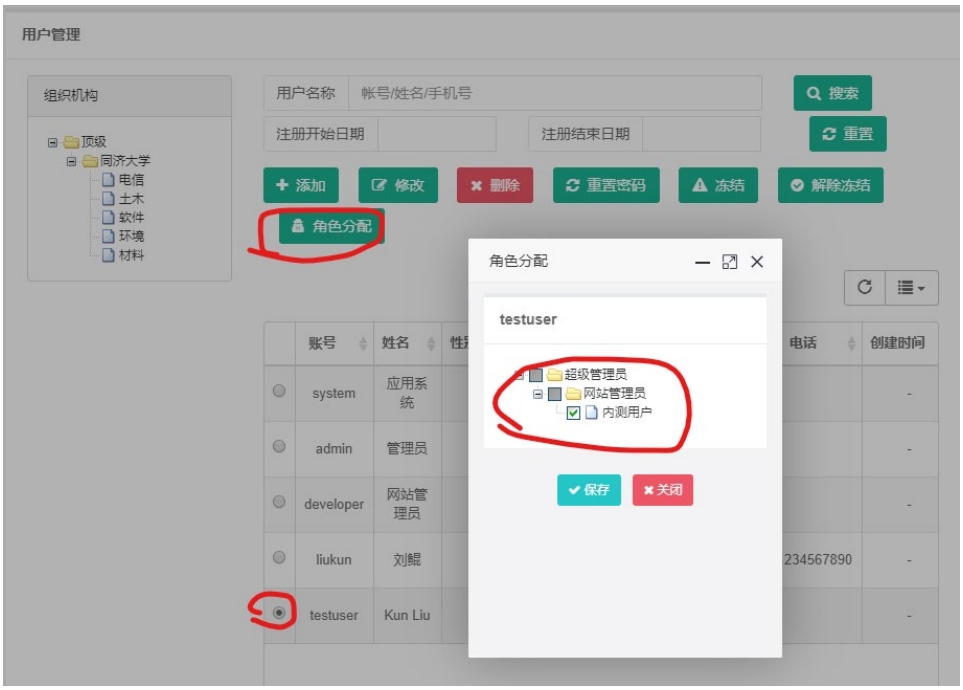
点击“添加”，输入用户信息：



提交后，结果如下：

说明成功添加用户。

选中新增加的用户，然后点击角色分配：



就可以分配内测用户给新用户。

	账号	姓名	性别	角色	部门	邮箱	电话	创建时间
○	system	应用系统		-	-			-
○	admin	管理员	女	超级管理员	土木学院	eniluzt@qq.com		-
○	developer	网站管理员	男	网站管理员	软件学院	eniluzt@qq.com		-
○	liukun	刘鲲	男	内测用户	电信学院	123456@qq.com	1234567890	-
○	testuser	Kun Liu	男	内测用户	材料学院			-

可以看到，角色分配成功。

重新登录这个新分配的用户，操作都正常。

7. 课设中遇到的问题及解决

7.1. 通过 Navicat 导出数据字典

一个一个写 excel 文档很累，但我们可以使用查询导出数据字典：

```
USE information_schema;

SELECT
    C.COLUMN_NAME AS '字段名',
    C.COLUMN_TYPE AS '数据类型',
    C.IS_NULLABLE AS '允许为空',
    C.EXTRA AS 'PK',
    C.COLUMN_COMMENT AS '字段说明'
FROM
    COLUMNS C
INNER JOIN TABLES T ON C.TABLE_SCHEMA = T.TABLE_SCHEMA
AND C.TABLE_NAME = T.TABLE_NAME
WHERE
    T.TABLE_SCHEMA = '数据库名' and T.TABLE_NAME='表名'
```

7.2. Invalid task '.test.skip=true'

在打包时，maven 会进行测试。而经常运作是正常的，但打包的时候出现了问题。所以其实可以跳过检查阶段：

mvn package -Dmaven.test.skip=true

但是这时候包标题的错误，经过查找资料，发现原因在于这个减号。需要用一个`声明这个是指令用的：

mvn package `-Dmaven.test.skip=true

7.3. MySQL 8.0 远程连接改权限与 5.7 不同

首先，需要在阿里云打开 3306 端口，否则是不行的。

输入访问的 host 和密码，报 2059 错误，这是因为 MySql8.0 版本 和 5.7 的加密规则不一样。

出现这个原因是 MySQL 8 之前的版本中加密规则是 mysql_native_password,而在 mysql8 之后,加密规则是 caching_sha2_password.

解决问题方法有两种,一种是升级 navicat 驱动，一种是把 MySQL 用户登录密码加密规则还原成 mysql_native_password.

这里采用第二种方式 ：

修改加密规则:

ALTER USER 'root'@'%' IDENTIFIED BY 'password' PASSWORD EXPIRE NEVER;

password 为当前密码。

更新 root 用户密码:

ALTER USER 'root'@'%' IDENTIFIED WITH mysql_native_password BY 'password';

password 为新设置的密码。

刷新权限:

FLUSH PRIVILEGES;

设置完成，再次使用 Navicat 连接数据库：

7.4. 项目部署到服务器上要加项目名

在 VSCode 上运行的时候，默认是省略项目名的，所以地址栏为：“localhost:8080/login.html”，但是，项目部署到服务器上后，是把项目的 war 包放到 tomcat/webapps 目录下，然后直接运行的，所以必须要加项目名，访问地址应为：“服务器 IP 地址:8080/项目名/login.html”。

7.5. Tomcat 无法正常关闭

使用./shutdown.sh 关闭 Tomcat,有时会关闭成功,有时会出现关闭错误。

这时候，查看 catalina.out 日志记录，使用（tail 指令，不需要 cat 打开），发现出现

Connection refused 错误。

org.apache.catalina.startup.Catalina stopServer

SEVERE: Could not contact localhost:8005. Tomcat may not be running.

org.apache.catalina.startup.Catalina stopServer

SEVERE: Catalina.stop:

java.net.ConnectException: Connection refused (Connection refused)

at java.net.PlainSocketImpl.socketConnect(Native Method)

at java.net.AbstractPlainSocketImpl.doConnect(AbstractPlainSocketImpl.java:350)

at java.net.AbstractPlainSocketImpl.connectToAddress(AbstractPlainSocketImpl.java:206)

at java.net.AbstractPlainSocketImpl.connect(AbstractPlainSocketImpl.java:188)

at java.net.SocksSocketImpl.connect(SocksSocketImpl.java:392)

at java.net.Socket.connect(Socket.java:589)

at java.net.Socket.connect(Socket.java:538)

at java.net.Socket.<init>(Socket.java:434)

at java.net.Socket.<init>(Socket.java:211)

at org.apache.catalina.startup.Catalina.stopServer(Catalina.java:498)

at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)

at sun.reflect.NativeMethodAccessorImpl.invoke(NativeMethodAccessorImpl.java:62)

at sun.reflect.DelegatingMethodAccessorImpl.invoke(DelegatingMethodAccessorImpl.java:43)

at java.lang.reflect.Method.invoke(Method.java:498)

at org.apache.catalina.startup.Bootstrap.stopServer(Bootstrap.java:343)

at org.apache.catalina.startup.Bootstrap.main(Bootstrap.java:430)

由于一方面操作被拒绝，一方面 tomcat 没有启动开造成程序无法访问。

解决办法是：使用 ps -ef | grep tomcat 指令，查看与 tomcat 相关的进程，kill 掉。

继续查看 Tomcat 是否关闭,如出现以下信息则说明 Tomcat 已关闭

root 3044 2838 0 20:20 pts/1 00:00:00 grep --color=auto tomcat

7.6. **双向关联表查询时异常：**java.lang.StackOverflowError: null

报错的原因是栈溢出。是由于使用 Jpa 查询时产生了死循环或无限递归。

比如新系统中两个双向关联的实体列：

在 User 中，我这样写：

```
public class User    extends BaseEntity {
    @ManyToMany(targetEntity = Subject.class)
    @JoinTable(name = "user_subject", joinColumns = @JoinColumn(name =
"user_id"),
```

```
        inverseJoinColumns = @JoinColumn(name = "subject_id"))
        private Set <Subject> subjects = new HashSet<>();
    }
```

而在 subjects 中：

```
public class Subject extends BaseEntity {
    @ManyToMany(mappedBy = "subjects")
    private Set<User> users;
}
```

在遍历集合时，

```
for（Subject e : user.getSubjects()）{
     System.out.println(e);
}
```

输出一个 Subject 对象时，也会输出 users，因为是双向关联的，所以也会触发 user 对象，接下来便是 subjects……产生无限递归，导致栈溢出。

解决方法一种是：

```
for（Subject e : user.getSubjects()）{
        e.setUsers(null); //提前终止，这样就不会无限递归
        System.out.println(e);
}
```

另一种解决方法是：

把双向的关联变为单向的关联，也就是单向多对多，关系只由一方维护。

我这里是只让 User 维护，即删除 Subject.java 中的：

```
@ManyToMany(mappedBy = "subjects")
private Set<User> users;
```

于是系统恢复正常。

8. 课程设计总结

在之前，我没有体验过使用 Java 进行工程项目的开发。而在本次课程设计之中，我看了《Java 核心技术》,《设计模式 Head First》，对设计模式和 Java 有了一定的了解。

而决定使用 Java 进行开发，则是因为在找实习，发现阿里对于 Java 的需求很大，而我们课程的安排里没有对于 Java 有大量运用的课程，没充足的训练。我把这次课设当作一个可以拿来找实习的项目来做。

一开始，我花了大量的时间了解框架。我发现以前流行的框架如 SSM 已经不流行了，现在流行 Spring 全家桶。我顿时感到恐惧：技术更新的迭代实在是太快了。我了解到的 Spring 的依赖注入。这种方法大大降低了程序的耦合性，真的是太妙了，真是伟大的发明

创造，Java 工程师的福音。

我曾经看到网上有教授说，框架只是一时的，而设计模式是基本不变的。我们应该花80%的时间学习、体会设计模式，20%的时间学框架。

在开发的过程中，Java 相关的错误实在太多，我收集了一大堆问题、教程及其解决。一些文档和教程更新太慢，或者是错的，互相抄来抄去，比如 MySql 的 8.0 和之前的版本在用户密码设置上很不一样，我花了很多时间设置阿里云服务器数据库的远程登录，纠结了很久；部署服务器、打包工程项目也是处处是坑。有时候一个 bug 找了一天是正常的事。

不过，工欲善其事必先利其器，我先是花了半天的时间找到了 Spring Boot 的热部署方法，然后之后的测试和开发就特别快，因为我不需要每次重启以应用变换。而且我使用了 Visual Studio Code 进行开发而不是一些常见的笨重的 Java IDE，也改善了开发体验。

网上有一个段子，说 Java 程序员就是 Chrome 开几十个标签页，所以需要占用大量内存。我也切实体会到这一点了。

再说到这个论文推送系统的 idea，也是有一天突发奇想。原本我没什么好的想法，想做一个简单的项目练手，但后来来了灵感，于是找了一个开源框架，实现了这个科研资讯推送系统。我认为这个系统还很幼稚，但 idea 是很好的，并且没有太多同质项目。

总之，我在这次项目中收获很多，体验了完整的开发流程，增长了非常多的知识。尤其是对于阿里云的申请、访问、打包、配置等等，可以说肯定对我以后的帮助特别大。

新系统未来的发展目标是，将爬虫这一步骤搬到服务器上，通过脚本自动每日更新。而这也需要更多的 Linux 的知识，还有很长的路要走。

9. 参考文献

[1] Cay S. Horstmann. Java 核心技术[M]. 北京: 机械工业出版社, 2016.

[2] Freeman, E. el. Head First 设计模式(中文版)[M]. 北京: 中国电力出版社, 2007.

[3] Spring 实战（第五版）[EB/OL]. https://potoyang.gitbook.io/spring-in-action-v5/, 2019/2020-05-01.

[4] http://enilu.gitee.io/guns-lite

[5] 使用 Navicat 连接阿里云服务器上的 MySQL 数据库[EB/OL]. https://blog.csdn.net/liuhailiuhai12/article/details/64124637, 2006/2020-05-02.

[6] ALLCONFERENCES.COM[EB/OL]. http://www.allconferences.com, 2002/2020-05-05.

[7] 夕小瑶的卖萌屋[EB/OL]. http://arxiv.xixiaoyao.cn, 2020/2020-05-06.

[8] Spring Data JPA 中的一对一，一对多，多对多查询[EB/OL]. https://super-aviator.github.io/2019/06/22/, 2018/2020-05-04.

[9] springboot 实现热部署[EB/OL]. https://blog.csdn.net/chachapaofan/article/details/88697452, 2019/2020-05-06.

[10]通过 Navicat 导出数据字典[EB/OL]. https://www.jianshu.com/p/9af2c1dc010f, 2019/2020-05-09.
[11]Invalid task '.test.skip=true' [EB/OL]. https://kuniganotas.wordpress.com/2011/08/12/invalid-task-test-skiptrue-you-must-specify-a-valid-lifecycle-phase/, 2011/2020-05-10.
[12]双向关联表查询时异常：java.lang.StackOverflowError: null[EB/OL]. https://blog.csdn.net/east123321/article/details/80050394, 2018/2020-05-07.