诚 信 声 明

我声明，所呈交的毕业论文是本人在老师指导下进行的研究工作及取得的研究成果。据我查证，除了文中特别加以标注和致谢的地方外，论文中不包含其他人已经发表或撰写过的研究成果，也不包含为获得其他教育机构的学位或证书而使用过的材料。我承诺，论文中的所有内容均真实、可信。

毕业论文作者签名： 签名日期： 年 月 日

师生科研互动平台

**[摘 要]**

目前，学生要参加各级创新竞赛、本科生研究项目或开放实验项目，与老师的交流不是十分畅通。本项目为电气信息学院研制基于Web的科研交流平台，针对学生的创新创业计划、开放实验项目等，老师或学生可以通过该平台实现交流，克服学生盲目选题或对选题理解不深等问题。

建立基于网络的师生科研信息交流和合作的互动平台十分必要。通过该平台，教师和学生发布相关科研信息，包括教师寻找学生参与的课题、各级创新项目的互动、挑战杯项目、开放实验项目以及毕业设计的互动等，实现学生和老师在相关项目和课题的网上交流与合作以及老师在线指导学生，讨论和答疑。

**[关键词]**

WEB；AJAX；NODE.JS；MONGO DB；JQUERY

**Abstract：**

/\*将中文摘要译成英文，建议在二稿时完成\*/

**Keywords：**  ； ；

/\*关键词是供检索用的主题词条，应采用能覆盖论文主要内容的通用技术词条，一般列3～5个，按词条的外延层次从大到小排列。关键词之间以“；”号间隔 \*/

目 录

[1 绪论 1](#_Toc418518052)

[1.1 文献综述 1](#_Toc418518053)

[1.2 研究框架 2](#_Toc418518054)

[1.2.1 研究目标 2](#_Toc418518055)

[1.2.2 研究内容 2](#_Toc418518056)

[1.2.3 创新之处 2](#_Toc418518057)

[1.2.4 研究方法 3](#_Toc418518058)

[1.3 术语说明 3](#_Toc418518059)

[1.3.1 Node.js 3](#_Toc418518060)

[1.3.2 Mongo DB 3](#_Toc418518061)

[1.3.3 AJAX 3](#_Toc418518062)

[1.3.4 jQuery 4](#_Toc418518063)

[1.3.5 Bootstrap 4](#_Toc418518064)

[2 设计环境 5](#_Toc418518065)

[2.1 硬件环境 5](#_Toc418518066)

[2.2 软件环境 5](#_Toc418518067)

[3 设计方案 6](#_Toc418518068)

[3.1 技术路线 6](#_Toc418518069)

[3.1.1 服务端技术 6](#_Toc418518070)

[3.1.2 数据库技术 7](#_Toc418518071)

[3.1.3 前端技术 7](#_Toc418518072)

[3.2 功能模块设计 8](#_Toc418518073)

[3.2.1 用户模块 8](#_Toc418518074)

[3.2.2 项目模块 9](#_Toc418518075)

[3.2.3 团队模块 9](#_Toc418518076)

[3.2.4 评论模块 9](#_Toc418518077)

[3.3 数据库设计 9](#_Toc418518078)

[3.3.1 用户表 10](#_Toc418518079)

[3.3.2 项目表 11](#_Toc418518080)

[3.3.3 团队表 12](#_Toc418518081)

[3.3.4 选课表 12](#_Toc418518082)

[3.3.5 入团申请表 13](#_Toc418518083)

[3.3.6 评论表 13](#_Toc418518084)

[3.4 架构设计 13](#_Toc418518085)

[4 路由设计 15](#_Toc418518086)

[4.1 Express简介 15](#_Toc418518087)

[4.2 路由入口 15](#_Toc418518088)

[4.3 具体路由示例 18](#_Toc418518089)

[4.3.1 页面GET请求 18](#_Toc418518090)

[4.3.2 页面POST请求 19](#_Toc418518091)

[5 重点模块设计 21](#_Toc418518092)

[5.1 用户模块 21](#_Toc418518093)

[5.1.1 注册控制 21](#_Toc418518094)

[5.1.2 登录控制 22](#_Toc418518095)

[5.1.3 功能控制 22](#_Toc418518096)

[5.2 项目模块 23](#_Toc418518097)

[5.2.1 项目权限 24](#_Toc418518098)

[5.2.2 项目加入 24](#_Toc418518099)

[5.2.3 项目更新 24](#_Toc418518100)

[5.2.4 项目分享 25](#_Toc418518101)

[5.3 评论模块 25](#_Toc418518102)

[5.3.1 直接评论 25](#_Toc418518103)

[5.3.2 回复评论 25](#_Toc418518104)

[6 平台效果图 26](#_Toc418518105)

[6.1 PC浏览器端效果图 26](#_Toc418518106)

[6.2 移动设备浏览器端效果图 29](#_Toc418518107)

[结论 32](#_Toc418518108)

[致谢 33](#_Toc418518109)

[参考文献 34](#_Toc418518110)

# 绪论

研制开发本平台的目的，主要是方便学生参加各级创新竞赛、本科生研究项目或开放实验项目，以及方便学生与教师在整个项目过程中的交流，克服学生盲目选题或对选题理解不深问题。

本文总结了研制开发本平台的整个设计与实现过程，技术路线，遇到的问题与解决方案等，在设计与开发过程中将软件工程基本理念、设计模式等于实际技术相结合，除实现基础功能以外，在平台的健壮性、安全性、伸缩性、易用性等方面都做了充分的研究与扩展。

## 文献综述

Web开发截止到现在，可以分为两个历史阶段：Web 1.0和Web 2.0。Web1.0基本采用的是技术创新主导模式，信息技术的变革和使用对于网站的新生与发展起到了关键性的作用。进入Web 2.0以后，Web开发不只限于对开发语言与框架等技术层面的追求，对于用户体验和系统性能也要求的越来越高，最为突出的表现是数据的海量增加。

Web前端技术的发展是推动Web平台的一大重要源动力。进入 Web2.0时代以后，服务器端在处理大数据与高并发方面承受着巨大的压力，于是Web前端开始承担越来越多的责任。同时人们已经不满足于Web界面原始的展现方式。UI，用户体验，可复用性开始成为受关注的焦点。随着需求的愈来愈强烈，各种RIA（Rich Internet Applications）也如同雨后春笋一般发展起来，其中最主要的系别是基于JavaScript的RIA库。

开放是互联网的发展的趋势，为了顺应“开放”这一潮流，越来越多的数据、资源、服务在网络服务平台上被发布出来，不但丰富了网络应用的功能，也带来了一个又一个创新的商业模式。同时，为了实现网络的开放化，诸多的新技术、新标准被引入到互联网应用的平台建设中。

## 研究框架

### 研究目标

本设计将建立基于Web的师生科研信息交流和合作的互动平台，该平台针对学生的创新创业计划、开放实验项目等，通过该平台，教师和学生发布相关科研信息，包括教师寻找学生参与的课题、各级创新项目的互动、挑战杯项目、开放实验项目以及毕业设计的互动等，实现学生和老师在相关项目和课题的网上交流与合作以及老师在线指导学生，讨论和答疑。

### 研究内容

包括Web平台整体架构的设计，服务端技术的选用及设计，数据库技术的选用及设计，前端技术的选用及设计，以及对Web平台安全、性能、伸缩、易用等方面的研究实现等。

### 创新之处

响应式设计已成为Web界的热门话题，考虑到用户群的特点，平台除提供通过常规计算机浏览器使用的方式以外，还将同时为移动设备（如移动电话、平板电脑等）设计界面，充分利用HTML5以及CSS3的高级特性，在保证其在各大浏览器下表现一致的前提下，使其成为一个跨设备的应用。同时考虑到我国实际情况，将尽量保持平台对较低版本浏览器的兼容性。

在服务端选择了较新的Node.js技术以及非关系型数据库来进行开发，同时对平台的安全方面做充分的考虑，将是一项充满挑战的任务。

### 研究方法

对已有技术主要通过阅读开发文档以及开发案例等方法进行学习与使用，在完成基础功能的同时结合课堂学习的软件工程知识，将软件工程思想应用到平台的每一个角落。

## 术语说明

### Node.js

基于开源Google Chrome V8引擎的JavaScript运行环境，旨在构建快速、可伸缩的网络应用。轻量级、高效的Node.js使用事件驱动以及非阻塞的IO模型，适用于数据密集型的、实时型的、跨平台的应用。[1]

### Mongo DB

Mongo DB是一个开源免费的基于分布式文件存储的数据库，旨在为WEB应用提供可扩展的高性能数据存储解决方案。Mongo DB是一个介于关系数据库和非关系数据库之间的产品，他支持的数据结构非常松散，是类似JSON的BSON格式，因此可以存储比较复杂的数据类型。Mongo最大的特点是他支持的查询语言非常强大，其语法类似于面向对象的查询语言，几乎可以实现类似关系数据库单表查询的绝大部分功能，而且还支持对数据建立索引。[2]

### AJAX

AJAX（Asynchronous JavaScript And XML）是指一种创建交互式网页应用的网页开发技术。通过在后台与服务器进行少量数据交换，AJAX 可以使网页实现异步更新。这意味着可以在不重新加载整个网页的情况下，对网页的某部分进行更新。传统的网页（不使用 AJAX）如果需要更新内容，必须重载整个网页页面。

### jQuery

jQuery是一个开源免费的兼容多浏览器的JavaScript库，也是如今是最流行的JavaScript库，在世界前10000个访问最多的网站中，有超过55%在使用jQuery。jQuery的语法设计可以使开发者更加便捷，其模块化的使用方式使开发者可以很轻松的开发出功能强大的静态或动态网页。

### Bootstrap

Bootstrap，来自 Twitter，是目前最受欢迎的开源免费前端框架。它由Twitter的设计师Mark Otto和Jacob Thornton合作开发，是一个CSS/HTML框架。Bootstrap提供了优雅的HTML和CSS规范，一直是GitHub上的热门项目，包括NASA的MSNBC（微软全国广播公司）的Breaking News都使用了该项目。

# 设计环境

## 硬件环境

|  |  |
| --- | --- |
| 计算机 | ACPI x64-based PC |
| 处理器 | Intel® Core™ i5-4570 CPU @ 3.20GHz |
| 主板 | Dell 0WMJ54 |
| 硬盘 | ST1000DM003-1ER162 ATA Device |
| 内存 | 8GB Hynix DDR3 1600MHz |
| 显卡 | Intel HD Graphics 4600 |
| 网卡 | RTL8168/8111/8112 Gigabit Ethernet Controller |

## 软件环境

|  |  |
| --- | --- |
| 操作系统 | Windows 7 professional x64 SP1 |
| 数据库 | Mongo DB v3.0.2 |
| 运行环境 | Node.js v0.10.36 |
| 开发工具 | Brackets Release 1.1 build 1.1.0-15558 |
| 调试工具 | Chrome @ 42.0.2311.90 m Developer Tools |

# 设计方案

前期设计是否优秀、合理是软件成败的关键所在，本章节主要对平台的技术选用、功能模块设计、数据库设计、架构设计等方面进行分析与论述。

## 技术路线

### 服务端技术

Web2.0时代，性能是永恒的主题。举例说，我们暨南大学的综合教务管理系统，功能方面非常完备，能够很好地完成各种各样的教学日常任务，是非常优秀的Web1.0时代产物。但是到了今天，其短板也暴露无遗，最主要的体现就是每到全校选课时段，系统就会因为无法承受高并发带来的压力而效率大减，甚至崩溃，使着急选课的同学感到非常无奈。在互联网愈来愈普及的今天，如何处理高并发带来的服务器压力是一项炙手可热的话题，其方法主要有两个：一是提高硬件基础；二则是从软件方面着手。在硬件条件有限的情况下，软件技术的选择和使用就成为了非常重要的环节。

因此，在选择Web平台的服务端技术时，性能、效率、成本、发展等因素都需要经过慎密的考虑与求证，最终得到适合于平台特点与需求的答案。在对比研究Java，Microsoft .NET，PHP，Node.js等技术后，最终选择了Node.js作为平台的服务端开发技术。

Node.js 是一个服务器端的JavaScript环境。Node用谷歌公司开发的V8引擎运行JavaScript。借助于V8，Node得以提供一个服务器端的运行环境来快速的编译和执行JavaScript语言。它速度的提升主要是因为V8把JavaScript编译为本地机器代码（直接解释执行），而不是编绎为字节码或中间代码（再解释执行）。Node.js是开源的，跨平台的，能够运行在各种操作系统上。Node.js的核心是事件循环机制。为了处理超大规模的并发请求，Node.js中所有的I/O集中操作都是异步的、非阻塞的。作为一种较新的技术，目前网络上有很多针对Node与其它服务端技术在IO与REST性能方面的对比，结论是其远超如Java，PHP等的传统技术。这意味着在相同的硬件条件下，使用Node的Web应用可以处理更多的客户端请求。

### 数据库技术

随着Web2.0的兴起，传统的关系数据库在应付超大规模和高并发的Web2.0纯动态网站已经显得力不从心，暴露了很多难以克服的问题，而非关系型的数据库则由于其本身的特点得到了非常迅速的发展。NoSQL（Not Only SQL）数据库的产生就是为了解决大规模数据集合多重数据种类带来的挑战，尤其是大数据应用难题。

结合Node的特点，决定选用开源、免费的文档型NoSQL数据库Mongo DB作为平台数据库，这样得以搭建一个全栈的JavaScript Web应用框架（Mongo DB使用基于JavaScript的Shell进行操控）。

### 前端技术

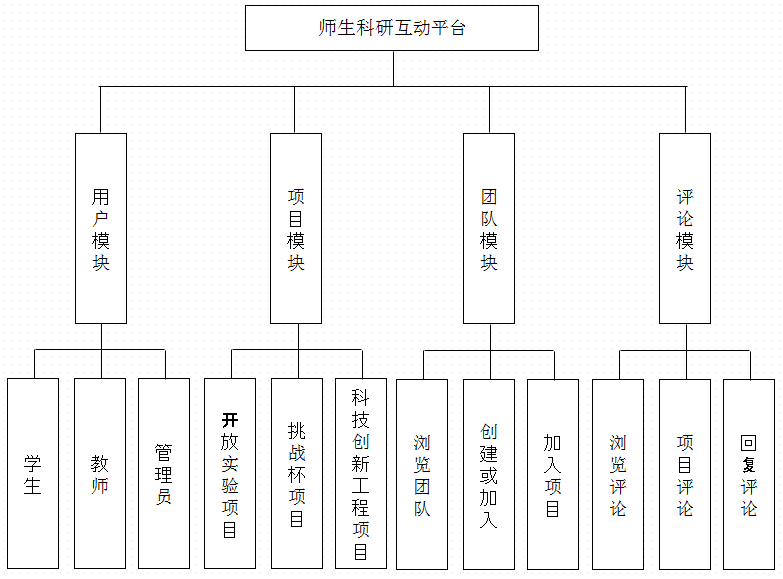
在Web前端选用较常规的jQuery库作为JavaScript支持，同时选用Twitter Bootstrap作为HTML/CSS基本架构，搭配其它开源支持项目进行开发。本平台在Web前端的主要特点是充分利用了HTML5与CSS3的新特性，除提供通过常规计算机浏览器使用的方式以外，还将同时为移动设备（如移动电话、平板电脑等）设计界面，使其成为一个跨平台、跨设备的云应用。

在Web前端与服务端的交互方面，全部采用AJAX的方式进行数据交互。这样做的好处是将页面展示与后台逻辑完全解耦，在减轻服务器负担的同时，也更加符合软件工程理念的基本思想，以及便于扩展开发和维护。

## 功能模块设计

在学习项目任务书以及跟指导教师讨论以后，确定了以下的一些主要功能模块，见图2-1

图 ‑1



### 用户模块

主要包括用户的注册控制、登录控制以及管理控制等。同时，学生和教师角色可以在个人中心查看自己拥有或申请加入的项目、团队，或查看其他人对自己的回复评论，管理自己的个人信息（姓名、班级等）以及密码。管理员角色可以对平台的所有信息进行管理，拥有最高权限，使用独立的登录以及注册模块。

### 项目模块

平台的初步设计实现阶段主要涉及三种项目：开放实验项目、挑战杯项目以及科技创新工程项目。其中，开放实验项目只允许由教师开设，学生可以独立选课。挑战杯项目、科技创新工程项目允许由教师开设或学生申请并寻找指导教师，学生必须组成团队后方可以由团队负责人进行选课。

### 团队模块

包括团队的浏览、寻找以及创建、加入、选课等功能。团队只能由学生创建，创建者为负责人，其它学生以及教师可以申请加入。挑战杯项目、科技创新工程项目需要以团队为单位选课，选课团队必须有指导教师。

### 评论模块

用户可以对各种项目以评论、回复等方式进行交流，未登录用户可以查看评论，登录后可以发表评论。回复目标将可在其个人中心查看到对其的所有回复评论。

## 数据库设计

由于使用的是NoSQL数据库，在数据库的设计上会与关系数据库有一些不同之处，其一是数据表的主键将全部由Mongo DB在数据插入时自动生成的Object ID担任。但为了保持平台数据的一致性与完整性，将沿袭关系数据库的一些设计方法。

### 用户表

表 ‑1

|  |  |  |  |
| --- | --- | --- | --- |
| 键名 | 类型 | 属性 | 备注 |
| Id | String | 非空，唯一 | 用户的学号或工号，作为登录名 |
| Password | String | 非空 | 用户密码，以MD5码储存 |
| Key | String | 非空 | 随机生成的秘钥，以MD5码储存 |
| Name | String | 非空 | 用户姓名 |
| Sex | String | 枚举 | 用户性别：男、女或其它 |
| Type | String | 枚举 | 用户类型：同学、老师或管理员 |
| Active | Boolean | 非空，默认为false | 未激活用户不能登录，用于注册审核及毕业后失效 |
| Email | String |  | 用户邮箱地址 |
| Phone | String |  | 用户电话号码 |
| Student Attr | Object |  | 此对象用于储存“同学”类用户的特有属性，包括学院、年级、专业等 |
| Teacher Attr | Object |  | 此对象用于储存“老师”类用户的特有属性，包括部门、职称等 |

### 项目表

表 ‑2

|  |  |  |  |
| --- | --- | --- | --- |
| 键名 | 类型 | 属性 | 备注 |
| Name | String | 非空 | 项目名 |
| Type | String | 非空，枚举 | 项目类型：开放实验项目、挑战杯项目或科技创新工程项目 |
| Description | String | 非空 | 项目简介 |
| College | String | 非空 | 项目所属学院 |
| Date Start | Date | 非空 | 项目预计开始时间 |
| Date End | Date | 非空 | 项目预计结束时间 |
| Date Update | Date | 非空 | 项目最后更新时间 |
| Teacher | Object Id | 外键 | 项目指导教师 |
| Creator | Object Id | 外键 | 项目创建者 |
| Open Experiment Attr | Object |  | 此对象用于储存“开放实验项目”的特有属性，包括容量、学时、实验室等 |
| Challenge Cup Attr | Object |  | 此对象用于储存“挑战杯项目”的特有属性，包括类型、基础、目标等 |
| Innovation Project Attr | Object |  | 此对象用于储存“科技创新工程项目”的特有属性，包括摘要、关键词等 |

### 团队表

表 ‑3

|  |  |  |  |
| --- | --- | --- | --- |
| 键名 | 类型 | 属性 | 备注 |
| Name | String | 非空 | 团队名 |
| Leader | Object Id | 外键 | 团队负责人 |
| Description | String | 非空 | 团队描述 |
| Date Create | Date | 非空，默认为创建的时间 | 团队创建时间 |

### 选课表

表 ‑4

|  |  |  |  |
| --- | --- | --- | --- |
| 键名 | 类型 | 属性 | 备注 |
| Student | Object Id | 外键 | 选课学生 |
| Team | Object Id | 外键 | 选课团队 |
| Project | Object Id | 外键 | 所选项目 |
| Active | Boolean | 非空，默认为false | 选课激活状态（是否通过审核） |
| Date | Date | 非空，默认为创建的时间 | 选课时间 |

### 入团申请表

表 ‑5

|  |  |  |  |
| --- | --- | --- | --- |
| 键名 | 类型 | 属性 | 备注 |
| User | Object Id | 外键 | 申请学生 |
| Team | Object Id | 外键 | 申请团队 |
| Active | Boolean | 非空，默认为false | 申请激活状态（是否通过审核） |
| Date | Date | 非空，默认为创建的时间 | 申请时间 |

### 评论表

表 ‑6

|  |  |  |  |
| --- | --- | --- | --- |
| 键名 | 类型 | 属性 | 备注 |
| From | Object Id | 非空，外键 | 评论用户 |
| To | Object Id | 外键 | 评论回复对象 |
| Project | Object Id | 外键 | 评论项目 |
| Body | String | 非空 | 评论内容 |
| Date | Date | 非空，默认为创建的时间 | 评论时间 |

## 架构设计

架构设计是软件开发的重要环节，本平台由于使用的服务端技术需要实现HTTP服务器，所以客户端请求的路由设计成为了架构设计的重点。整体设计见表2-7

表 ‑7

|  |
| --- |
| **表现层**：EJS模板，jQuery，Bootstrap |
| **路由层**：Express框架及其支持库 |
| **标准层**：Node.js技术及其支持库 |
| **数据持久层**：Mongoose Mongo DB ODM |
| **基础层**：操作系统，Mongo DB数据库 |

# 路由设计

由于网站需要实现HTTP服务器的客户端请求过程以及处理客户端AJAX请求，所以路由设计非常重要，直接影响到整个平台的结构以及性能。本章节主要介绍平台的路由系统设计。

## Express简介

Express 是一个简洁、灵活的 Node.js Web 应用开发框架，对Node的HTTP模块进行了高度封装。它提供一系列强大的特性、丰富的 HTTP 快捷方法和任意排列组合的 Connect 中间件，能够达到相当于Node原生应用的性能。使用Express框架主要的好处是快速开发和易于维护。

## 路由入口

代码 ‑1

|  |
| --- |
| // app.js  var express = require('express'),  session = require('express-session'),  mongoose = require('mongoose'),  path = require('path'),  favicon = require('serve-favicon'),  logger = require('morgan'),  cookieParser = require('cookie-parser'),  bodyParser = require('body-parser'),  moment = require('moment'),  app = express();  mongoose.connect('mongodb://localhost/test');  moment.locale('zh-cn');  app.set('views', path.join(\_\_dirname, 'views'));  app.set('view engine', 'ejs');  app.use(favicon(\_\_dirname + '/public/favicon.ico'));  app.use(logger('dev'));  app.use(bodyParser.json());  app.use(bodyParser.urlencoded({  extended: true,  limit: 100000  }));  app.use(session({  secret: '666',  resave: false,  saveUninitialized: true  }));  app.use(cookieParser());  app.use(express.static(path.join(\_\_dirname, 'public')));  app.use('/', require('./routes/index'));  app.use('/api/get', require('./routes/api/get'));  app.use('/api/post', require('./routes/api/post'));  app.use(function (req, res, next) {  var err = new Error('Not Found');  err.status = 404;  next(err);  });  if (app.get('env') === 'development') {  app.use(function (err, req, res, next) {  res.status(err.status || 500);  if (err.status === 404) {  res.render('404', {  user: req.session.user  });  } else {  res.render('error', {  message: err.message,  error: err  });  }  });  }  module.exports = app; |

代码说明：

该文件代码是单个客户端请求进入服务器后的整个处理及路由过程，即平台程序的直接入口。每个客户端请求都会经历路由的整个过程，最后由平台做出相应的响应。

以下是对主要代码的详细说明。

代码 ‑2

|  |
| --- |
| var express = require('express'),  …  app = express(); |

这段代码的作用是加载依赖的Node模块，并启动Express应用。

代码 ‑3

|  |
| --- |
| mongoose.connect('mongodb://localhost/test'); |

使用Mongoose连接本地数据库。

代码 ‑4

|  |
| --- |
| app.set('views', path.join(\_\_dirname, 'views'));  app.set('view engine', 'ejs'); |

设置表现层页面的路径，并设置页面引擎为EJS渲染引擎。

代码 ‑5

|  |
| --- |
| app.use(bodyParser.json());  app.use(bodyParser.urlencoded({  extended: true,  limit: 100000  }));  app.use(session({  …  })); |

将请求的Post数据（如果存在）转换为JSON格式，并限制大小为100KB。设置Session。

代码 ‑6

|  |
| --- |
| app.use('/', require('./routes/index'));  app.use('/api/get', require('./routes/api/get'));  app.use('/api/post', require('./routes/api/post'));  app.use(function (req, res, next) {  var err = new Error('Not Found');  err.status = 404;  next(err);  }); |

路由主体，按顺序监听了三个路径，分别是'/'、'/api/get'与'/api/post'，其请求具体处理逻辑包含在相应的文件中。若请求路径不在其中，则抛出404异常。

## 具体路由示例

### 页面GET请求

代码 ‑7

|  |
| --- |
| router.get('/center', function (req, res) {  if (req.session.user) {  if (req.session.user.type === '管理员') {  res.redirect('/admin');  } else {  res.render('center', {  user: req.session.user  });  }  } else {  res.redirect('/login');  }  }) |

这段代码接收了客户端对<http://hostname/center> URL的GET请求，并在处理前首先进行了判断：

1. 若已登录且登录类型为管理员，则重定向到管理员页面；
2. 若已登录且登录类型不是管理员，则渲染center页面并发送给客户端；
3. 若未登录，则重定向到登录页面。

### 页面POST请求

代码 ‑8

|  |
| --- |
| router.post('/signin', function (req, res) {  if (req.query.type === 'admin') {  model.User.findOne({  type: '管理员'  }, function (err, doc) {  …  if (md5(req.body.password + doc.key) === doc.password) {  req.session.user = doc;  res.sendStatus(200);  } else {  res.sendStatus(401);  }  …  });  } else {  …  model.User.findOne(condition).lean().exec(function (err, doc) {  if (err || !doc) {  res.sendStatus(401);  } else {  if (md5(req.body.password + doc.key) === doc.password) {  if (doc.active) {  …  req.session.user = doc;  res.sendStatus(200);  } else res.sendStatus(403);  } else {  res.sendStatus(401);  }  }  })  }  }) |

这段代码处理了客户端对<http://hostname/api/post/signin> URL的POST请求

1. 若请求类型为管理员，则从User表中取得管理员记录与用户输入密码比对，匹配则设置Session并返回200 OK，否则报错
2. 若请求类型为其它，则从User表中根据响应ID获取记录后与用户输入密码比对，匹配则设置Session并返回200 OK，否则报错

# 重点模块设计

本章节主要对平台的一些重点模块的设计与实现方法进行说明。

## 用户模块

考虑到用户在大部分行为以及管理上的一致性，并没有将用户根据类别（学生、教师、管理员）分开到不同的数据表存放，而是统一存入用户表，根据表中的类型属性来区分。

### 注册控制

控制流程：

1. 用户填写注册信息
2. 页面使用MD5算法对密码加密后使用AJAX提交数据
3. 后台对提交数据进行XSS过滤，并设置激活状态为FALSE
4. 后台生成随机秘钥，并使用该秘钥与提交的密码组合后再使用MD5算法加密，生成储存在数据库中的密码
5. 后台尝试储存该用户记录，若储存成功则返回200 OK，不成功（ID冲突）则返回500错误
6. 页面根据返回的HTTP状态码做出响应，告知用户注册成功，或ID已存在

详细代码：

代码 ‑1

|  |
| --- |
| user.active = false;  xssUser(user);  user.key = md5(new Date());  user.password = md5(user.password + user.key);  user = new model.User(user);  user.save(function (err) {  res.sendStatus(err ? 500 : 200);  }); |

### 登录控制

控制流程：

1. 用户填写登录名及密码
2. 页面使用MD5算法对密码加密后使用AJAX提交数据
3. 后台根据登录名获取正确密码与秘钥，比对后根据匹配结果判断是否登录成功，并返回响应的HTTP状态码
4. 页面根据返回的HTTP状态码做出响应，告知用户登录成功，或遇到了问题（密码不匹配，账号未激活等）

详细代码：请见代码4-8

### 功能控制

某些功能需要用户登录后才提供使用，如发布项目、发表评论等，某些功能只对管理员开放，如各种数据增删改等，对此平台主要从两个方面对此进行控制：一是在渲染页面时根据登录情况调整渲染元素，以及在页面脚本中加入是否登录的判断从而决定是否提交请求以及提醒用户登录（简称前端控制），二是在服务端对用户提交的请求根据Session情况判断是否作出响应（简称后台控制），这样做的好处是可以使平台的性能与安全性都得到了一定的保障。

详细代码：

1. 前端举例（教师或学生可以创建项目，学生不能创建开放实验项目）

代码 ‑2

|  |
| --- |
| <% if(typeof user!=='undefined' && user.type!=='管理员' && ((user.type==='老师' && projectType==='开放实验项目' ) || projectType==='挑战杯项目' || projectType==='科技创新工程项目' )){%>  <button …> … 创建</button>  <% } %> |

1. 后台举例（管理员可以获取所有用户的信息，已登录用户可以获取自己的信息，未登录用户不能获取用户信息）

代码 ‑3

|  |
| --- |
| router.get('/user', function (req, res) {  if (req.session.user) {  if (req.session.user.type === '管理员') {  …  } else {  if (req.query.self) {  res.status(200).send(req.session.user)  } else {  req.sendStatus(404);  }  }  } else {  res.sendStatus(401)  }  }) |

## 项目模块

同用户模块，没有将不同类型的项目划分到不同的表中储存，主要目的是便于管理以及扩展，若将来需要添加一种新的项目，只需要在数据库中为其增加一种类型即可，同时也无需求改增删改查方法。

为防止恶意提交，在创建一个项目时，项目所有者将被强制设定为当前Session的用户，且项目内容将会经过XSS过滤。

### 项目权限

1. 未登录用户可以查看项目列表，查看项目详细内容以及指导教师简要信息等
2. 已登录学生可以选课、提交报告、选择自己的团队参加项目，以及发布挑战杯项目、科技创新工程项目以寻求导师等
3. 已登录教师可以发布所有类型的项目，并处理学生以及团队对项目的申请

### 项目加入

学生可以申请加入开放实验项目，或者组建团队并以团队为单位申请加入挑战杯项目或科技创新工程项目。

学生提交申请后，教师可以在“个人中心”查看并处理（接受或拒绝）申请。对于挑战杯项目或科技创新工程项目，在接收一个团队后，其它团队将不能再加入该项目，并且其它团队对该项目提交的申请也会失效。

### 项目更新

项目所有者可以对项目进行内容的更新，学生发布的项目在指导教师加入后所有者即变更为该指导教师。

### 项目分享

项目详细页面添加了一系列分享按钮，提供了将页面分享到各大社交网络的接口，任何浏览者都可以参与页面分享，从而便于教师寻找学生加入项目，也便于学生寻找其它有兴趣的同学组建团队一同加入项目。该分享功能使用了“百度分享”的部分模块。

## 评论模块

为防止恶意提交，在提交一条评论时，评论的“来自用户”属性将被强制设定为当前Session的用户，且评论内容将会经过XSS过滤。代码如下。

代码 ‑4

|  |
| --- |
| req.body.from = req.session.user.\_id;  xssComment(req.body);  var comment = new model.Comment(req.body);  comment.save(function (err) {  res.sendStatus(err ? 500 : 200)  }) |

### 直接评论

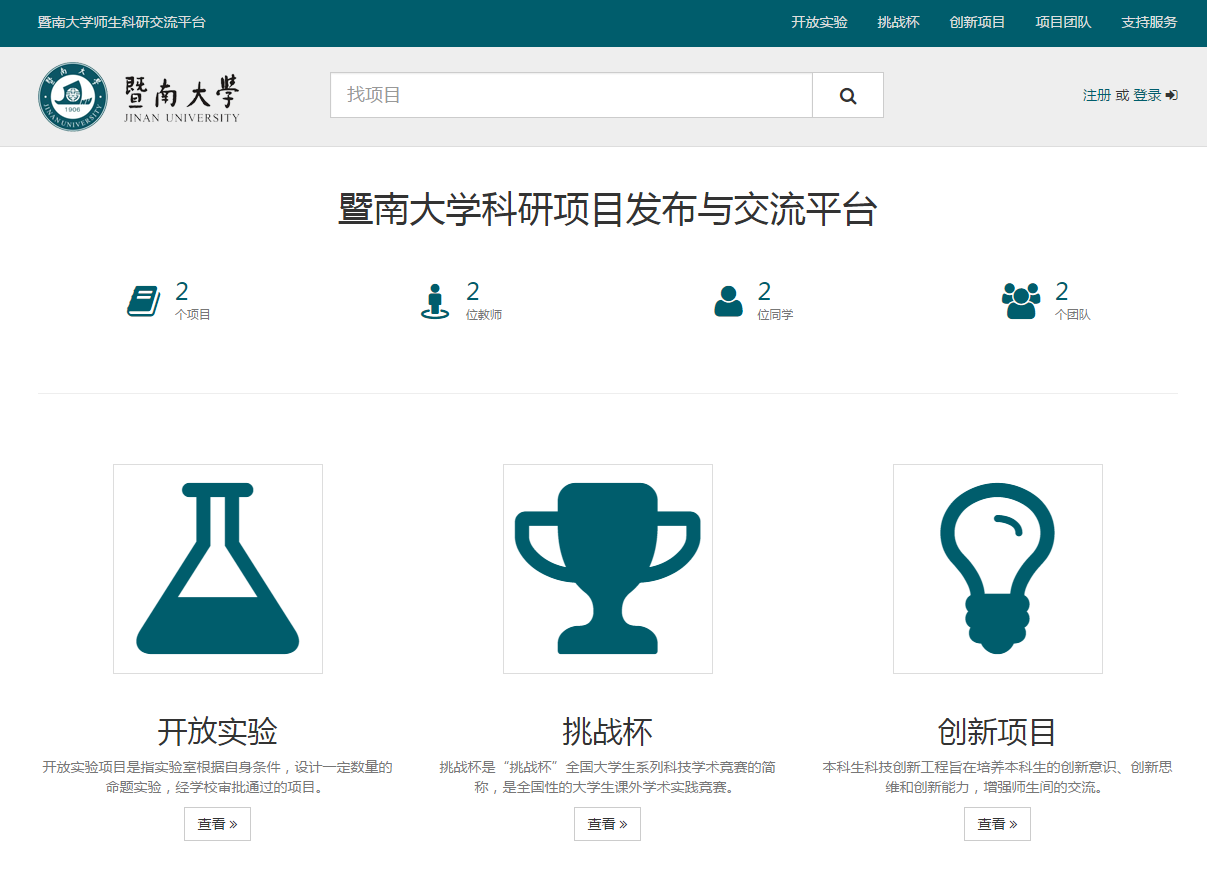
已登录用户可以在项目内发表自己对项目的评论以及想法等，是实现平台师生交流的重要方法。浏览者在打开该页后即可以查看历史评论内容。

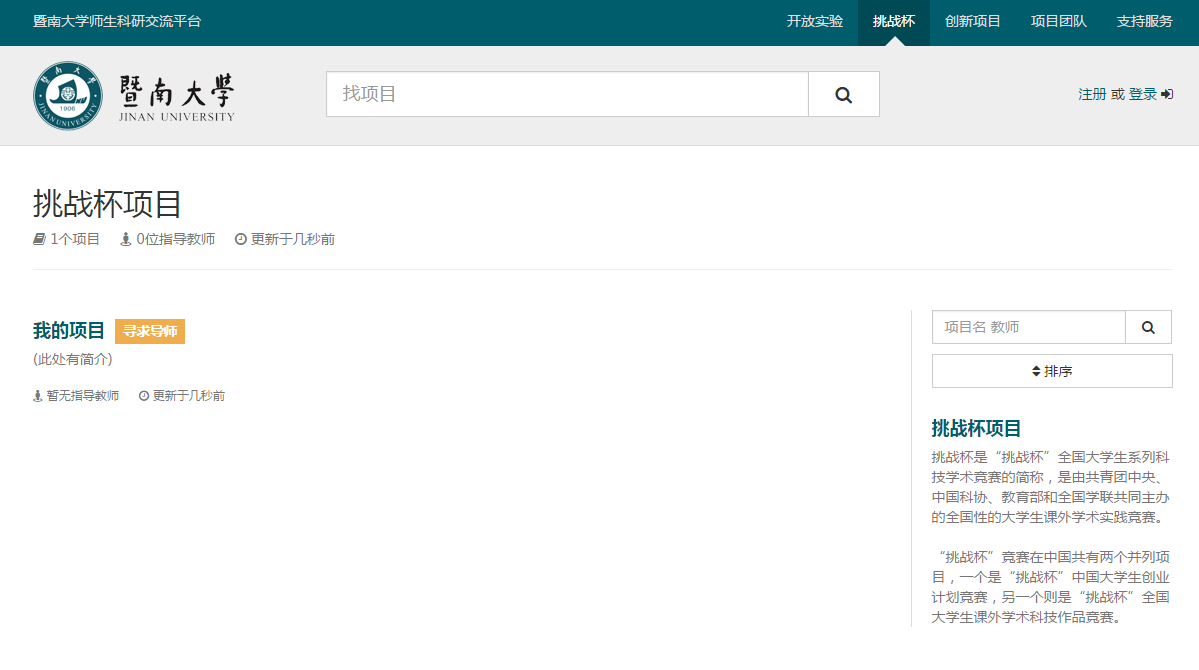
### 回复评论

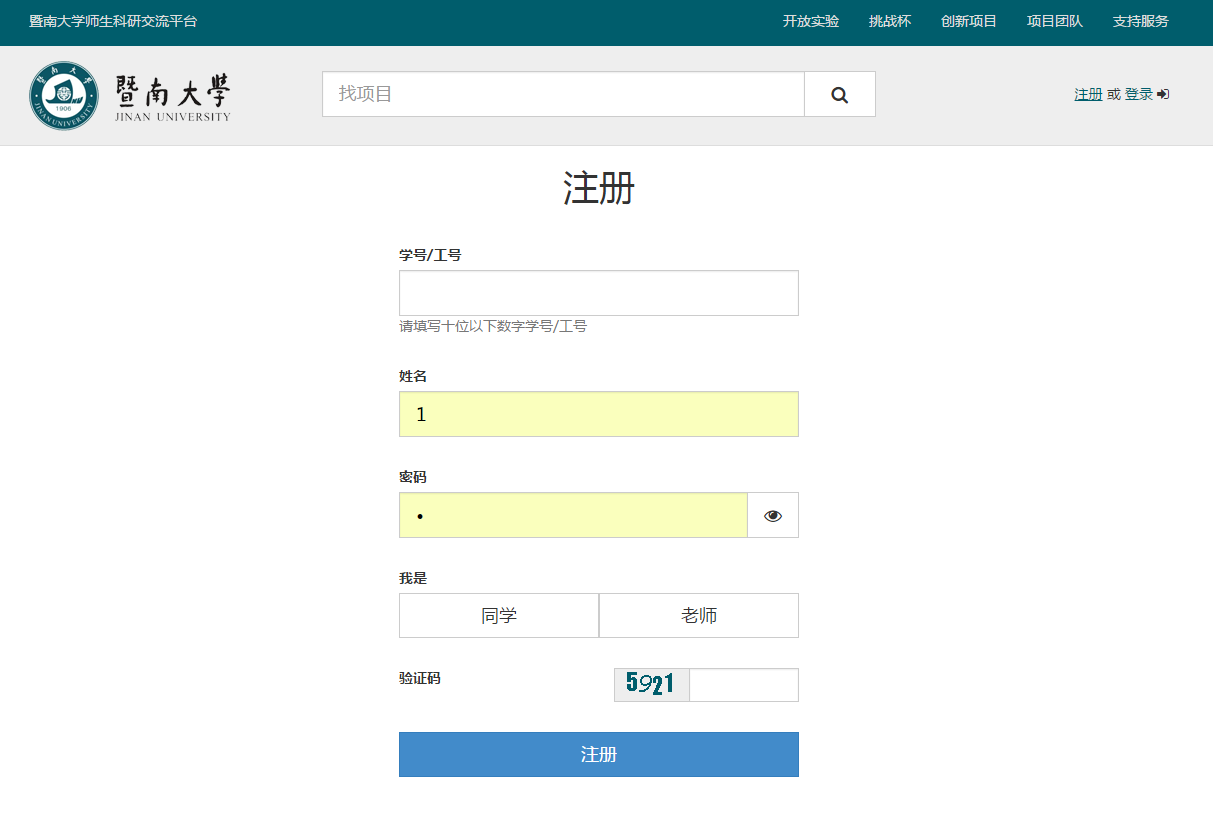
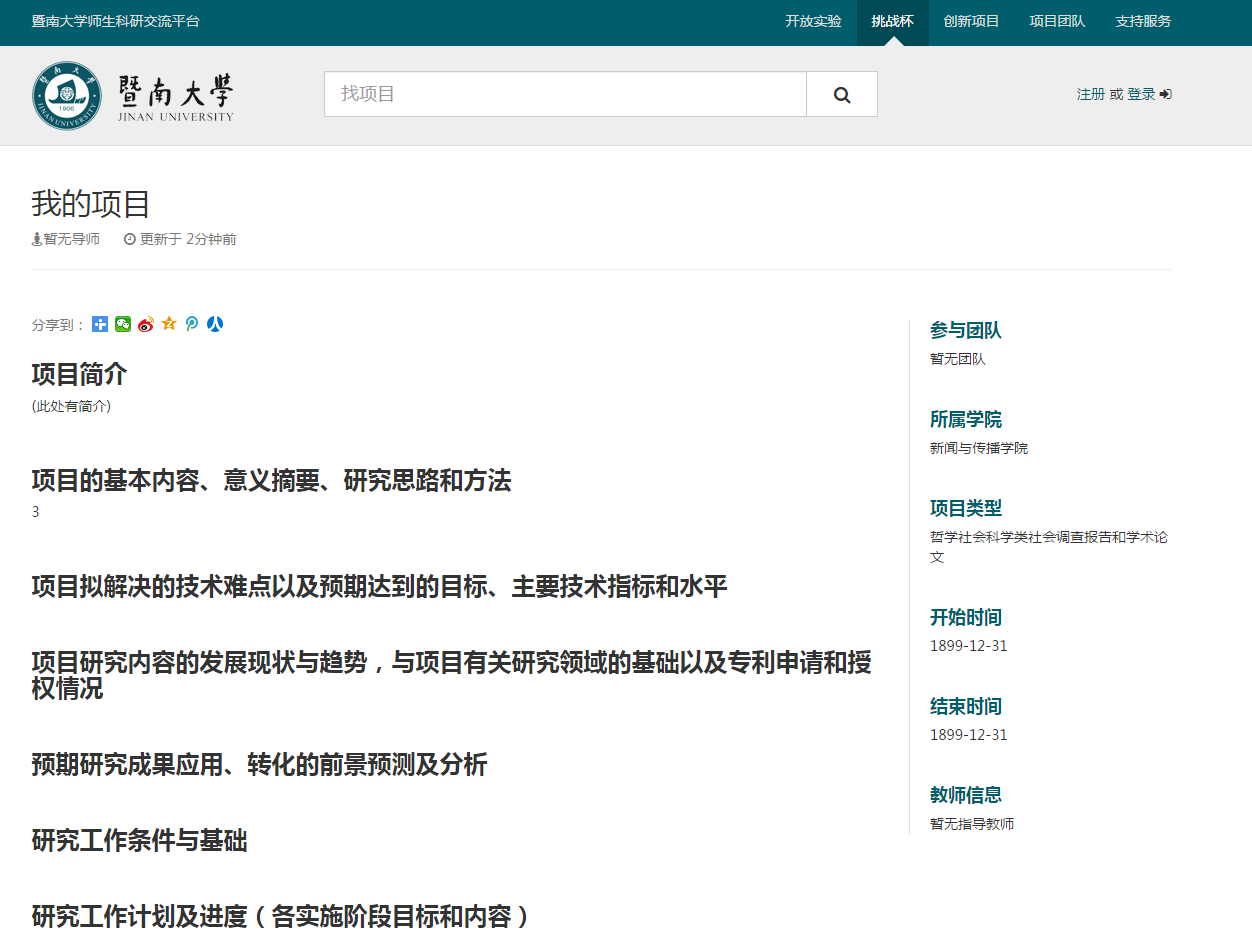
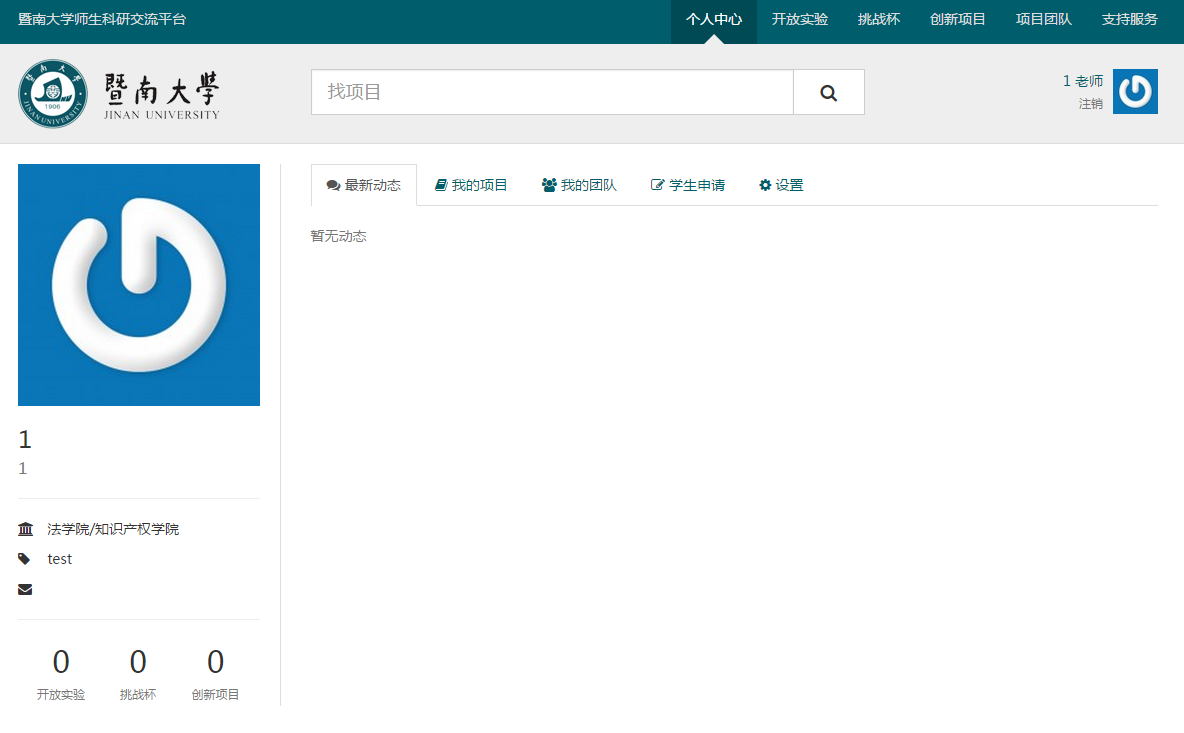
已登录用户可以针对某个用户的某条评论进行回复，同时用户可以在其个人中心发现其他用户对自己的回复。

# 平台效果图

## PC浏览器端效果图





## 移动设备浏览器端效果图



# 结论

《师生科研互动平台》是由潘冰老师开设，最终需要一个实用的网络平台并且挂载到学校服务器供学校师生使用的毕业设计项目。其第一阶段的功能需求并不复杂，但为了适应越来越多的移动设备以及现代网站的响应式设计趋势，网站的前端设计以及美化成为了重要的一环。同时，互联网技术的快速发展也使如今的建站技术不再限于传统的.Net，Java等，数据库技术也不限于传统的关系数据库。为了更好地学习以及利用新型技术，我最终选择了目前非常热门但年轻的Node.js以及Mongo DB作为平台的建站基础，放弃了更为成熟稳定的技术。以上两点是本平台的最大的创新之处。

在完成毕业设计期间，我遇到了许多问题，其中有自己翻阅资料解决的，有询问指导教师解决的，也有询问同学解决的。最终能顺利完成毕业设计，大家的帮助是非常重要的。

在我看来，毕业设计是对学生所学专业知识的一次综合检验。虽然在学校期间并没有开设过相关技术的课程，但是核心理念都是一致的，就是软件工程思想，软件架构设计与设计模式等。同时，Web编程技术的思想也是非常重要的，如何提高性能，如何防范攻击，如何提高用户体验等，都是完成一项Web应用最需要的知识。

# 致谢

感谢我的导师潘冰老师，谢谢他对我的悉心指导。在毕设选题以后我对题目有非常多的疑惑之处，潘冰老师对我一一进行了说明与指导，在技术选择方面也给予了关键的支持。

感谢我的班主任陈双平老师，在大学期间接收我作为ACM成员，虽然由于个人原因没有坚持到最后，但期间收获了非常多宝贵的知识。

感谢周密老师，开设的软件工程设计理念系列课程，是大学后半段时期中最重要的收获。

同时也感谢四年来教导过我的其它老师，感谢陪伴我走过大学生活的同学，感谢在背后支持我的父母。

毕业设计中使用到了大量开源框架及作品，参考了大量前人的经验、书籍等，可以说是站在巨人的肩膀上完成的，感谢他们的无私奉献。

# 参考文献

[1] Ryan Dahl. Node.js . [CP/OL]. (2015)[2015-05-04]. <https://nodejs.org/>.

[2] 张善友. 分布式文件储存的数据库开源项目MongoDB. (2010-05-20)[2015-05-04]. http://www.cnblogs.com/shanyou/archive/2010/05/20/1739987.html.

1、电子文献转载其他非电子文献（如电子图书、电子报刊），应在源文献的著录格式后著录电子文献的引用日期和获取和访问路径，其文献类型标志使用复合标志，即[文献类型标志/文献载体标志］。

[1] 江向东．互联网环境下的信息处理与图书管理系统解决方案[J/OL]．情报学报，1999，18(2)：4[2000-01-18] ．http://www.chinainfo.gov. cn/periodical/gbxb/gbxb99/gbxb990203．

2、 非第1种情况者使用下面著录格式：（注：联机文献中无出版地、出版者、出版年的可省略。）

［序号］主要责任者．题名[文献类型/载体类型］．出版地：出版者，出版年(更新或修改日期)[引用日期] ．获取和访问路径．

[21] 萧钮．出版业信息化迈人快车道[EB/OL] ．(2001-12-19)[2002-04-15] <http://www>. creader.com/news/20011219/200112190019.html．

附：参考文献著录中的文献类别代码

普通图书：M 会议录：C 汇编：G 报纸：N 期刊：J 学位论文：D 报告：R

标准：S 专利：P 数据库：DB 计算机程序：CP 电子公告：EB

 电子文献载体类型标志如下：磁带 MT，磁盘 DK，光盘 CD，联机网络 OL。