电子科技大学

University of Electronic Science and Technology of China

学 士 学 位 论 文

BACHELOR DISSERTATION

论文题目团队信息网络管理系统的设计与实现

学生姓名 曾路洋

学　　号 2010013060009

专　　业 通信工程

学　　院 通信与信息工程学院

指导教师 马立香

指导单位 电子科技大学

2014年5月27日

摘 要

随着互联网的飞速发展和社会信息化水平的不断提高，互联网在现代社会中的方方面面起着越来越重要的作用。同时，科研团队在信息管理方面，有着内部信息管理和对外交流展示的两大需求。因此，在本课题中设计并实现了一个团队网络信息管理系统，能提高科研团队内交流、管理的效率，并能通过对系统中数据的搜索、筛选、分析，给科研团队的决策提供数据上的支撑；同时利用这些数据，为科研团队的对外交流提供了一个展示的平台。

在本文中，首先介绍了在设计与实现本系统过程中使用到的相关技术基础。随后分析了选择这些技术来实现本系统的优点和原因。随后阐述了MVC架构的特点以及如何应用到Web应用中去。随后使用AJAX技术、PHP Yii框架以及MySQL数据库，设计了一个基于B/S结构的团队信息管理系统，并描述了它的整体架构和具体实现。最后进行对本系统进行了功能测试与性能测试，验证了本系统的设计与实现。

关键词：信息管理系统，B/S架构，动态网站，PHP，MySQL

ABSTRACT

With the rapid development of the Internet and the elevating progress of the informationization in soceity, the Internet has been gaining its influnce in nearly every aspect in modern society. Meanwhile, research groups in universities have two major demands in informtaion management: internal information management and external information display and communication. Therefore, in this project, a Informtaion Management System for research groups is designed and implemented, which can improve the effeciency in intercommunion and management in research groups, and can ultilize these data to be searched, filtered, and analysed, providing statistical support for decision making in research groups; meanwhile, these data are used to provide a platform for external communication and display.

In this thesis, technologies used in designing and implementing this system is firstly discussed. Then details about why and how those technologies are chosen is analysed. Then the features of MVC architecture and how to use MVC in the web application are discussed. After that, using AJAX, PHP, and the Yii framework, a information management system based on B/S architecture and its overall structure is designed, and its implementation details are provided. Finally, functional test and performance test are both conducted, verifying the the design and implementation of the system.

Keywords: Informtaion Management System, B/S Framework, Dynamic Website, PHP, MySQL

目 录

第1章 引言

1.1 课题题背景及意义

随着互联网技术的兴起，越来越多的信息管理、办公自动化、内容管理系统采用B/S架构实现，即浏览器-服务器架构。是Web兴起后的一种网络结构模式，Web浏览器是客户端最主要的应用软件。这种模式统一了客户端，将系统功能实现的核心部分集中到服务器上，简化了系统的开发、维护和使用。客户机上只要安装一个浏览器，服务器安装Oracle、Sybase、Informix、MySQL或SQL Server等数据库，浏览器通过Web Server 同数据库进行数据交互。B/S最大的优点就是可以在任何地方进行操作而不用安装任何专门的软件。只要有一台能上网的电脑就能使用，客户端零维护。系统的扩展非常容易。它具有跨平台、分布性特点，业务扩展简单、维护方便。

随着传统管理信息系统的功能复杂性不断越来越高，以及用户对系统易用性、易操作性要求不团提高，传统B/S模式架构的局限性越来越明显。Ajax(Asynchronous JavaScript and XML)[1]技术的发展为解决这种局限性的方法指明了一个方向。通过Ajax技术，采用B/S架构的程序也能在客户端电脑上进行部分处理和刷新，从而大大的减轻了服务器的负担；并增加了交互性，能进行局部实时刷新，能够达到和传统C/S（客户端-服务端）架构相同的用户体验。在本课题中拟通过设计一个团队网络信息管理系统，来契合这两大需求，提高科研团队内交流、管理以及科研的效率，并能通过对系统中的数据的搜索、筛选、统计、分析，给科研团队的决策提供数据上的支撑。

1.2 论文组织结构

本文一共分为六章。本章介绍了团队信息网络管理系统的设计与实现课题背景和意义。

第二章 理论基础：介绍了设计和实现本系统所需要的理论知识与技术基础。

第三章 需求分析：对了本系统需要实现的功能进行了分析。

第四章 方案设计：分析了本系统设计过程中的选择各种技术的原先以及如何使用这些技术来完成本系统的设计。

第五章 详细设计与实现：阐述了本系统详细设计与实现的过程，包括了：数据库实现、模型层实现、视图层实现以及控制层实现。

第六章 系统测试：对本系统进行了功能测试和性能测试，并针对测试结果进行分析。

第七章 结束语：总结与展望本课题。

第2章 技术基础

本章主要介绍在本系统的设计与实现过程用会用到的一些技术、软件工具以及设计模式。

2.1 B/S结构

B/S（Browser/Server）结构[2]即浏览器和服务器结构。它是随着Internet技术的兴起，对C/S结构的一种变化或者改进的结构。在这种结构下，用户工作界面是通过WWW浏览器来实现，极少部分事务逻辑在前端（Browser）实现，但是主要事务逻辑在服务器端（Server）实现，形成所谓三层结构。

2.2 AJAX技术

AJAX即“Asynchronous JavaScript and XML”（异步的JavaScript与XML技术），指的是一套综合了多项技术的浏览器端网页开发技术。Ajax的概念由Jesse James Garrett所提出[1]。

传统的Web应用允许用户端填写表单（form），当提交表单时就向Web服务器发送一个请求。服务器接收并处理传来的表单，然后送回一个新的网页，但这个做法浪费了许多带宽，因为在前后两个页面中的大部分HTML码往往是相同的。由于每次应用的沟通都需要向服务器发送请求，应用的回应时间依赖于服务器的回应时间。这导致了用户界面的回应比本机应用慢得多。

与此不同，AJAX应用可以仅向服务器发送并取回必须的数据，并在客户端采用JavaScript处理来自服务器的回应。因为在服务器和浏览器之间交换的数据大量减少（大约只有原来的5%）,服务器回应更快了。同时，很多的处理工作可以在发出请求的客户端机器上完成，因此Web服务器的负荷也减少了。

2.3 面向对象程序设计

在本系统的设计与实现过程中，用到了许多面向对象程序设计的思想与方法，故在这里对其进行介绍。

面向对象程序设计（Object-oriented programming，缩写：OOP）是一种程序设计范型，同时也是一种程序开发的方法。对象指的是类的实例。它将对象作为程序的基本单元，将程序和数据封装其中，以提高软件的重用性、灵活性和扩展性[3]。下面介绍了面向对象程序设计中的一些概念：

1. 类：定义了一件事物的抽象特点。通常来说，类定义了事物的属性和它可以做到的（它的行为）。
2. 对象：是类的实例。
3. 继承：是指，在某种情况下，一个类会有“子类”。
4. 多态：是指由继承而产生的相关的不同的类，其对象对同一消息会做出不同的响应。
5. 方法：也称为成员函数，是指对象上的操作，作为类声明的一部分来定义。方法定义了可以对一个对象执行那些操作。

2.4 LAMP

LAMP是指一组通常一起使用来运行动态网站或者服务器的自由软件名称首字母缩写：

1. Linux，操作系统
2. Apache，网页服务器
3. MySQL，数据库管理系统
4. PHP，脚本语言

2.4.1 服务器操作系统Linux

Linux作为本系统服务器的操作系统，主要有开源免费、良好的生态系统、更好的性能几点优势。

Linux 内核源代码可以免费下载。大多数Linux 发布版本，包括GNU/Linux 的发行版本和商业的发行版本几乎都提供免费下载服务。免费意味着零试用成本，也不需要为安装在第二台机器上付费。

Linux 作为服务器的优势是，他目前具有最好的生态系统，服务器端的各种软件都为它而设计，默认都认为你是在Linux上运行，Apache、MySQL等软件随能够在Windows下运行，但是性能却显著地比在Linux下运行时低[4]。

2.4.2 网页服务器Apache

Apache HTTP Server（简称Apache）是Apache软件基金会的一个开放源代码的网页服务器，可以在大多数计算机操作系统中运行，由于其跨平台和安全性被广泛使用，是最流行的Web服务器端软件之一[5]。它快速、可靠并且可通过简单的API扩充，将Perl、PHP、Python等解释器编译到服务器中。

2.4.3 数据库管理系统MySQL

MySQL是一个开放源代码的关系数据库管理系统（RDBMS），MySQL性能高、成本低、可靠性好，已经成为最流行的开源数据库，因此被广泛地应用在Internet上的中小型网站中。

2.4.4 脚本语言PHP

PHP（全称：PHP：Hypertext Preprocessor，即“PHP：超文本预处理器”）是一种开源的通用计算机脚本语言，尤其适用于网络开发并可嵌入HTML中使用[6]。PHP的语法借鉴吸收了C语言、Java和Perl等流行计算机语言的特点，易于一般程序员学习。PHP的主要目标是允许网络开发人员快速编写动态页面，但PHP也被用于其他很多领域。

2.4.4.1 PHP框架

PHP框架提供了一个用以构建web应用的基本框架，从而简化了用PHP编写web应用程序的流程。换言之，PHP框架有助于促进快速应用开发，不但节省开发时间、有助于建立更稳定的应用，而且减少了重复编码的开发。通过确保适当的数据库交换和在表现层编码，框架还可以帮助初学者建立更稳定的应用服务。这可以让你花更多的时间去创建实际的Web应用程序，而不是花时间写重复的代码。

PHP框架的作用相当于模型-视图-控制器（Model View Controller）。MVC是种编程的架构模式，将业务逻辑从UI中分离出来，允许一个一个单独修改（也称

为关注点分离)。在MVC中，Model指数据，View指表现层，Controller则指应用程序或业务逻辑。基本上，MVC打破了一个应用的开发进程，这样各组件就可以不受影响地各自工作。从本质上讲，这使得用PHP编码更快更简单。

2.4.4.2 Yii框架

Yii 是一个基于组件、用于开发大型Web 应用的高性能PHP 框架。它将Web 编程中的可重用性发挥到极致，能够显著加速开发进程。Yii代表简单(easy)、高效(efficient)、可扩展(extensible)。Yii框架的的特点：

1. Yii是一个纯OOP 框架。对于想使用Yii 的开发者而言，熟悉面向对象编程(OOP)会使开发更加轻松。
2. Yii 是一个通用Web 编程框架，能够开发任何类型的Web 应用。它是轻量级的，又装配了很好很强大的缓存组件，因此尤其适合开发大流量的应用，比如门户、论坛、内容管理系统(CMS)、电子商务系统，等等。
3. Yii 以性能优异、功能丰富、文档清晰而胜出其它框架。它从一开始就为严谨的Web 应用开发而精心设计，不是某个项目的副产品或第三方代码的组合，而是融合了作者丰富的Web 应用开发经验和其它热门Web 编程框架（或应用）优秀思想的结晶。

2.5 HTTP协议

在本系统中，使用了HTTP协议的GET方法和POST方法中的查询字符串（Query String）来完成递客户端与服务端的数据交互，在此对这些概念作简要的介绍。

超文本传输协议（HTTP）的设计目的是保证客户机与服务器之间的通信。HTTP 的工作方式是客户机与服务器之间的请求-应答协议[7]。

在客户机和服务器之间进行请求-响应时，两种最常被用到的方法是：GET和POST。

- GET ：从指定的资源请求数据。在GET方法中，查询字符串（键/值对）是在GET 请求的URL 中发送的，例如：index.php?name1=value1&name2=value2

- POST ：向指定的资源提交要被处理的数据，查询字符串（键/值对）是在POST 请求的HTTP 消息主体中发送的，例如：

  POST index.php HTTP/1.1
  Host: example.com
  name1=value1&name2=value2

第3章 系统需求分析

本章主要完成了整个系统的需求分析，主要包括了各个模块需要实现的功能，只有明确了需求，才能正确地实现本系统，对整个系统具有决策性、方向性、策略性的作用。

3.1 总体需求

为科研团队开发、设计一个“团队信息网络管理系统”，目标是建立及时、准确、全面的科研团队信息管理平台。通过与系统使用者进行沟通与调研，完成了本系统的需求分析。

本系统的目的针对科研团队信息管理的实际情况，全面覆盖科研团队在论文、科研项目、专利、人员管理等多个方面，并提供对这些数据的搜索、筛选以及分析功能，提高科研团队的工作以及沟通效率，并且为科研团队的决策、考核提供有力的支撑。同时，本系统需要利用这些数据，将论文、科研项目、专利等作为内容对外展示，方便其他人对团队进行了解和交流。

3.1.1 功能性需求

本系统的基本功能是实现科研团队信息的对外展示和内部管理，围绕这个功能，可将本系统的用户分为两种角色，一种角色是游客，另一种是本系统的管理员。游客只能访问每个数据管理模块的对外展示功能；而系统管理员可以对各项数据进行管理。因此，本系统应具有以下管理功能：

1. 管理员可以对各项数据进行录入、删除、修改、查询。
2. 管理员可以对数据进行多条件搜索、筛选。
3. 管理员可以对数据进行格式化地导入和导出。
4. 管理员可以设置
5. 游客可以查看各项数据对外展示的部分

另外考虑到在系统使用过程中，会有多个管理员对本系统所管理的数据进行维护，还应该进行对数据的分级权限管理，将管理员划分为超级管理员和子管理员，子管理员只能对某项或某几项特定的数据进行维护。

本系统的系统总体用例图如图 3–1 所示：

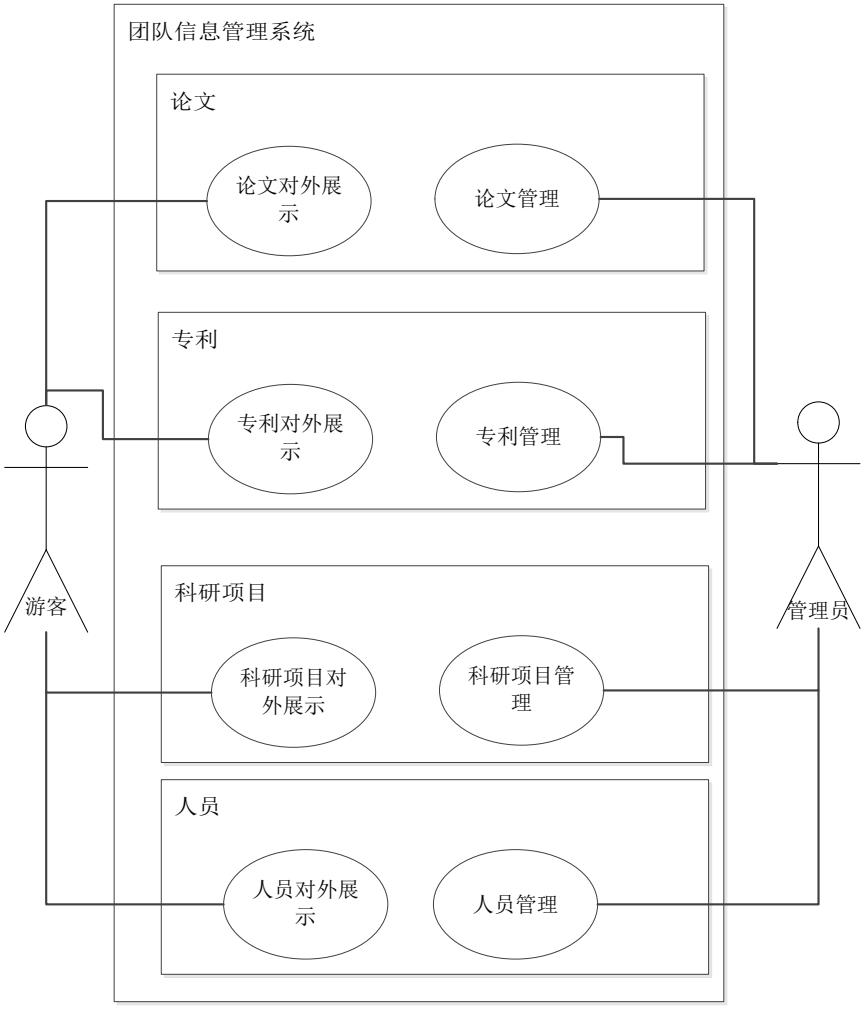

团队信息管理系统
论文
论文对外展示
论文管理
专利
专利对外展示
专利管理
科研项目
科研项目对外展示
科研项目管理
人员
人员对外展示
人员管理
游客
管理员

图 3–1 系统用例图

3.1.2 非功能性需求

本系统应该对用户来说满足易用，比如用户初次使用本系统时，应该能够批量的录入数据。

易用性的高低决定着用户对产品的第一印象是好是坏。因为本系统需要向用户展示大量的数据，于是用户需要一个简洁、易懂的界面，这样他们就能不需要任何帮助地一眼获得需要的信息。

另外，应该能够支持多终端对本系统的访问和使用，根据PC、平板电脑、智能手机等终端普通的屏幕大小和分辨率，采用不同的界面布局，提高系统的用户友好性。

考虑到科研团队的一般规模，应该能够支持200个用户的并发访问，以应对本系统多人同时使用的情形。另外，本系统在各种操作和处理中响应速度应为秒级甚至毫秒级，达到实时要求。尽量减少卡顿情况。

一个软件难免需要人来维护，或者需要对功能进行升级。很多时候，维护或升级的人员因为原先的代码可读性差以致无法顺利地进行进一步操作。因此，系统的程序源代码应该具备足够的易读性，具备标准的格式和简洁的代码风格，这样能使维护和升级顺利地进行。同时开发者还需提供详细的开发文档、部署文档以及使用说明。

限于篇幅所限，对于四个数据管理子模块，本文仅选取数据表项最为复杂、功能最多的论文管理模块和科研项目管理模块进行需求分析；专利管理模块以及人员管理模块数据表项与功能相对简单，且在功能上与论文管理模块和科研项目管理模块相类似，在本文中，仅给出它们的数据表项。

3.2 论文管理需求

通过将科研团队已发表和拟发表的论文录入到本系统中进行管理，可以及时地跟踪论文的发表状态，准确地根据作者、时间段、类别、支撑项目等不同条件搜索、筛选以及导出特定的论文，供科研团队里面的老师和学生进行使用，为团队的决策、考核提供有力的数据支撑。

论文管理模块主要要实现以下几个功能。

3.2.1 数据表项

需要设计一个数据库表，存储以下论文数据：

1. 论文信息，如：“Xiaoyan Huang, Yuming Mao, Fan Wu, “Low Complexity Utility-based Scheduling Algorithm for Heterogeneous Services in OFDM Wireless Networks,” In: Proc. of ICCCAS 2009, San Jose, USA, vol 1, pp.48-52.”
2. 作者：按顺序存储第一作者至第五作者
3. 状态：录用待发、已发表或已检索
4. 时间：录用时间、发表时间和检索时间
5. 检索类型：SCI、EI和ISTP的检索号
6. 论文级别：一级、核心、其他刊物、期刊、会议、国际、高水平
7. 论文文件：存储论文对应的PDF或者Microsoft Word文档
8. 支柱项目：数个支柱项目
9. 报账项目：数个报账项目

3.2.2 录入

分别需要实现对论文数据的批量录入和逐个录入。

批量录入指用户通过上传指定格式的Microsoft Excel表格，批量地将论文数据导入到数据库中。在批量录入中，需要实现替换已有表项和对已有表项进行添加的功能。

逐个录入：通过一个表单，提示用户输入或者选择以下内容：

1. 论文维护人员：从人员模块获取人员的姓名，生成下拉菜单进行选择，也可以通过输入拼音首字母快速选择
2. 论文信息
3. 作者1——作者5：与选择论文维护人员时相同
4. 状态：在“录用待发”、“已发表”、“已检索”中三选一
5. 时间：
    - 若状态为录用待发，则显示输入“录用时间”的表单
    - 若状态为已发表，则显示输入“发表时间”的表单
    - 若状态为已检索，则显示输入“发表时间”和“检索时间”的表单

6. 检索类型：仅当状态为已检索时，需要用户输入，提示用户输入SCI、EI或ISTP的检索号
7. 发表级别：在会议、期刊、国际、一级、核心、其他期刊中勾选一项或者多项
8. 支柱项目：在科研项目的列表中选择，也可以通过拼音首字母搜索
9. 报账项目：与选择支柱项目时相同
10. 高水平：在“是”、“否”中二选一
11. 论文文件：提示用户选择论文对应的pdf文件或者Microsoft Word文件进行上传

3.2.3 修改与删除

对已有表项进行修改与删除，能够先按条件搜索到需要修改或删除的论文，然后对相关项修改或删除。论文修改功能的用户界面与流程与论文录入的用户界面与流程一致，只是表单各条目需要显示已有数据。论文删除功能需要实现将选中的论文条目从存储论文的数据库表中删除。

3.2.4 显示

论文的显示功能需要分为对内显示和对外显示。

对外显示：仅选择类型为高水平的论文，具体功能如下：

1. 排序原则：最新日期置顶
2. 显示内容：数据表项中的论文信息
3. 显示方式：表格分页形式

对内显示：需要显示所有的论文，显示格式与对外显示一致。另外，需要在页面上醒目位置设置“查询”按钮，通过点击“查询”按钮，进入查询和导出功能。

3.2.5 查询与导出

需要可按单条件和组合条件查询，以表格形式显示并以Excel表格形式导出（显示和导出内容相同）。单条件包括：维护者、作者、状态、时间、检索类型、发表级别、支柱项目、报账项目。组合条件即是以上2个及以上条件的组合。

由于篇幅所限，本文在这里仅给出“按维护者查询”、“按作者查询”和“按时间段查询”的需求，其它条件、以及多条件查询的需求类似在这里给出的三种需求。

按维护者查询：排序按时间，最新时间置顶（录用、发表时间统一考虑），如时间相同，则以已检索、已发表、已录用再排。显示以及导出的格式如表 3–1 所示,其中：时间一项要求只录用的为录用时间，其它显示为发表时间；检索一项需要包含检索类型和检索号。导出的Excel文件名需要命名为“由XXX维护的论文”。

表 3–1 论文按维护者查询导出格式

序号	论文信息	状态	时间	检索
1	Xiaoyan Huang, Yuming Mao...	已检索	2011.01.01	EI:20111713930271
2	黄晓燕，毛玉明，吴凡，冷甦鹏.,“基于...	已发表	2009.05.01	

按作者查询：排序按时间，最新时间置顶（录用、发表时间统一考虑），如时间相同，则以已检索、已发表、已录用再排。导出的Excel文件名需要命名为“XXX发表的论文”。

按时间段查询：如2000年之后、2012年9月至2013年6月等，排序按时间，最新时间置顶（录用、发表时间统一考虑），如时间相同，则以已检索、已发表、已录用再排。导出的Excel文件名需要命名为“2000年至2013年之间发表的论文”。

3.3 科研项目管理需求

与论文管理模块类似，通过将科研项目的信息录入到本系统，可以及时地跟踪各个项目的当前状态和信息，按照人员、年份、级别等不同条件对科研项目进行搜索、筛选和显示，供科研团队里面的老师和学生进行使用，为团队的决策、考核提供有力的数据支撑。

3.3.1 数据表项

需要设计一个数据库表，存储以下信息：

1. 维护人员
2. 项目名称
3. 项目编号
4. 经本费编号
5. 级别：国际级、国家级、省部级、市级、校级、横向、国家自然基金、973、863、科技支撑计划、教育部高校博士点基金、重大专项和XX项目中的一项或者多项
6. 开始时间
7. 截止时间
8. 结题时间
9. 申报时间
10. 立项时间
11. 申报经费
12. 立项经费
13. 实际执行人员
14. 责任书人员：数个责任书人员

3.3.2 录入

同论文管理模块类似，分别需要实现对论文数据的批量录入和逐个录入。

批量录入指用户通过上传指定格式的Microsoft Excel表格，批量地将论文数据导入到数据库中。在批量录入中，需要实现替换已有表项和对已有表项进行添加的功能。

在逐个录入中，通过一个表单，提示用户输入或者选择以下内容：

1. 维护人员：从人员模块获取人员的数据，生成下拉菜单进行选择，也可以通过输入拼音首字母快速选择
2. 项目名称
3. 项目编号
4. 经本费编号

5. 级别：在国际级、国家级、省部级、市级、校级、横向、国家自然基金、973、863、科技支撑计划、教育部高校博士点基金、重大专项和XX项目中勾选一项或者多项
6. 时间：开始时间、截止时间、结题时间、申报时间、立项时间
7. 经费：申报经费、立项经费
8. 实际执行人员：在人员的列表中进行多项选择，也可以通过输入拼音首字母快速选择
9. 责任书人员：与实际执行人员的录入类似

3.3.3 显示

对外显示：要求校级项目和XX项目不显示，申报项目不显示。排序原则：项目级别和类型按时间排序（最新截至时间置顶），级别类型顺序如下级别类型从高到低：国际合作，国家自然基金，国家973，国家863，国家科技支持计划，重大专项，省部级，市级，横向。显示格式需要以表格的形式进行显示。

对内显示：研究项目分为责任版和执行版（主要是人员可能有差异）两个菜单可选。选择版本后，其显示栏目完全相同。栏目内容都显示。级别类型中，将XX项目放在最后。默认显示所有数据表项。对于申报项目，默认显示的数据表项：项目名称、级别、类型、申报时间、立项时间、人员。查询时置顶按相关条件。如按人员查询，则按其排名先后和时间后先（排名，最新时间），等等。

3.3.4 修改与删除

对已有表项进行修改与删除，能够先按条件搜索到需要修改或删除的科研项目，然后对相关项修改或删除。科研项目修改功能的用户界面与流程与科研项目录入的用户界面与流程一致，只是表单各条目需要显示已有数据。科研项目删除功能需要实现将选中的科研项目条目从存储科研项目的数据库表中删除。

3.3.5 查询

需要可按单条件和组合条件查询，以表格形式显示并以Excel表格形式导出（显示和导出内容相同）。单条件包括：维护者、实际参与人员、责任书参与人员、级别、时间等。组合条件即是以上2个及以上条件的组合。

3.4 其它数据管理需求

3.4.1 专利管理需求

专利管理模块的数据表项：

1. 维护人
2. 专利名称
3. 申请时间
4. 申请号
5. 授权时间
6. 授权号
7. 级别：国际或国内二选一
8. 类型：发明专利或实用新型专利二选一
9. 发明人：数个（少于五个）发明人
10. 报账项目

3.4.2 人员管理需求

1. 中文名
2. 英文名
3. 类型：学生或教师二选一
4. 介绍

3.5 权限管理需求

3.5.1 数据表项

需要设计一个数据库表，存储以下权限管理数据

1. 用户名
2. 密码散列值
3. 权限等级：系统超级管理员、论文模块管理员、科研项目模块管理员、专利管理员或人员管理员中的一个或者多个
4. 电子邮箱：用于忘记密码后的找回

3.5.2 权限的分级配置

在系统第一次部署时，需要提示当前用户（系统的部署者）输入一个电子邮箱地址，生成一个用户名为“admin”密码为随机生成的用户作为系统超级管理员，并将这些信息显示和发送给输入的电子邮箱地址。

有了系统超级管理员之后，管理员可以按需的查看、增加、删除、修改用户，并且为这些用户配置各个数据管理模块的管理权限。

3.5.3 用户管理

查看用户：按表格的形式列出当前系统所有用户，在每个用户条目位置显示增加、删除、修改的链接。

增加、修改用户：通过一个表单，提示管理员输入或选择以下内容：

1. 用户名：要求只能是英文字母和下划线的组合
2. 密码和重复密码：若密码和重复密码相符合，将密码的散列值存储在数据库中，不能存储明文密码，以免泄漏用户隐私
3. 权限等级：在系统超级管理员、论文模块管理员、科研项目模块管理员、专利管理员或人员管理员中的一个或者多个
4. 电子邮箱

第4章 系统方案设计

本章主要完成了整个系统的方案设计。首先分析了本系统在技术方案选择时的考虑，选择这些技术的原因以及好处；然后介绍了一种流行的面向对象设计模式，MVC设计模式；最后阐述了在本系统的设计中，如何利用MVC设计模式将各个模块解耦，实现在设计过程中的高内聚、低耦合，并给出了本系统的单次请求时序图，说明了本系统是如何运用MVC设计模式进行设计的。

4.1 技术选型

4.1.1 系统结构选择

对于本系统，可采用C/S结构或者B/S结构来实现。

4.1.1.1 传统C/S结构的缺点

C/S（Client/Server）结构，即大家熟知的客户机和服务器结构。它是软件系统体系结构，通过它可以充分利用两端硬件环境的优势，将任务合理分配到Client端和Server端来实现，降低了系统的通讯开销。

传统的C/S体系结构虽然采用的是开放模式，但这只是系统开发一级的开放性，在特定的应用中无论是Client端还是Server端都还需要特定的软件支持。由于没能提供用户真正期望的开放环境，C/S结构的软件需要针对不同的操作系统系统开发不同版本的软件，加之产品的更新换代十分快，已经很难适应百台电脑以上局域网用户同时使用。而且代价高，效率低。

其次，采用C/S架构，网络管理工作人员既要对服务器维护管理，又要对客户端维护和管理，这需要高昂的投资和复杂的技术支持，维护成本很高，维护任务量大。

4.1.1.2 B/S结构的优点

与C/S结构相比较，B/S结构的优点主要有跨平台性和易维护性[8]。

以目前的技术看，局域网建立B/S结构的网络应用，并通过Internet/Intranet模式下数据库应用，相对易于把握、成本也是较低的。它是一次性到位的开发，能实现不同的人员，从不同的地点，以不同的接入方式（比如LAN、WLAN、WAN,Internet/Intranet等）、不同的终端（PC机、智能手机以及平板电脑等）、不同的操作系统（Windows、Linux、Mac OS X、iOS、Android等）访问和操作共同的数据库；它能有效地保护数据平台和管理访问权限，服务器数据库也很安全。特别是在JAVA、PHP这样的跨平台语言出现之后，B/S架构管理软件更是方便、快捷、高效。

另外，软件系统的改进和升级越来越频繁，B/S架构的产品明显体现着更为方便的特性。对一个稍微大一点单位来说，系统管理人员如果需要在几百甚至上千部电脑之间来回奔跑，效率和工作量是可想而知的，但B/S架构的软件只需要管理服务器就行了，所有的客户端只是浏览器，根本不需要做任何的维护。无论用户的规模有多大，有多少分支机构都不会增加任何维护升级的工作量，所有的操作只需要针对服务器进行。

4.1.1.3 AJAX技术的优点

通过Ajax技术，采用B/S架构的程序也能在客户端电脑上进行部分处理和刷新，从而大大的减轻了服务器的负担；并增加了交互性，能进行局部实时刷新，能够达到和传统C/S结构相同的用户体验。

4.1.1.4 小结

综上所述，本系统选择采用基于AJAX技术的B/S结构，这样既能使本系统具有跨平台性和易维护性的优点，也能够提供和传统C/S结构相同的用户体验。

4.1.2 服务器软件选择

目前Internet上流行的网站构架方式是LAMP（Linux Apache MySQL PHP），即是用Linux作为操作系统，Apache作为Web服务器，MySQL作为数据库，PHP（部分网站也使用Perl或Python）作为服務器端脚本解释器。由于这四个软件都是开放源代码软件，因此使用这种方式可以以较低的成本创建起一个稳定、免费的网站系统。LAMP所有组成产品均是开源软件，是国际上成熟的架构框架，很多流行的商业应用都是采取这个架构，和Java/J2EE架构相比，LAMP具有Web资源丰富、轻量、快速开发等特点，微软的.NET架构相比，LAMP具有通用、跨平台、高性能、低价格的优势，LAMP软件可以说是当前最为流行的动态网页解决方案[9]。

因此，在本系统中，系统程序运行在Linux操作系统下，由Apache HTTP服务器提供内容，在MySQL数据库中存储内容，利用PHP来实现程序逻辑。

4.1.3 PHP框架选择

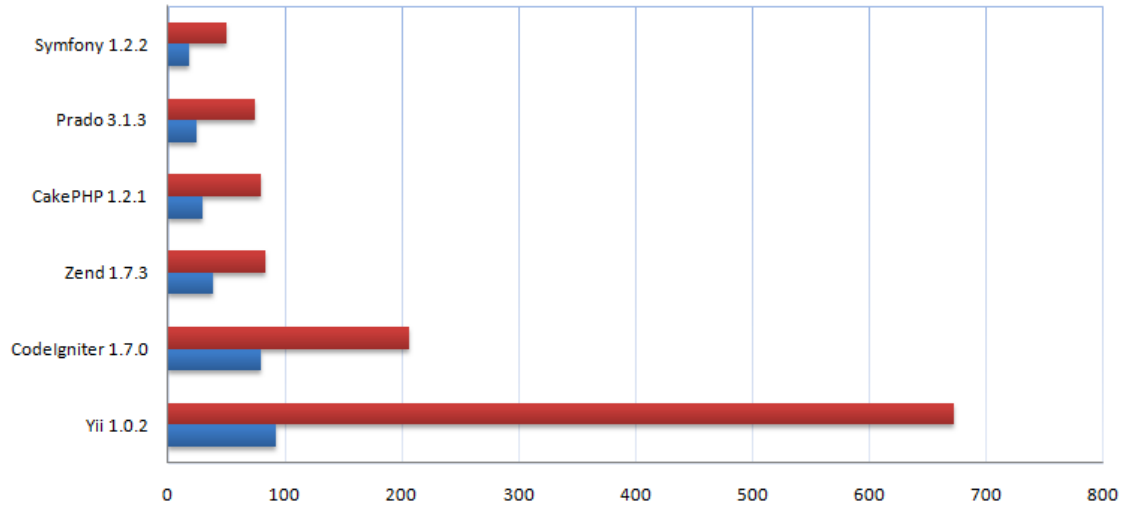
在图 4–1 中给出了 Yii 框架与其它主流框架的性能比较，其中PPS（Requets Per Second）指每秒钟请求数，反映了使用不同框架编写同样功能程序每秒钟所能处理的请求数目，越高越好；红色部分是没有开启APC（Alternative PHP Cache，是一个开放自由的PHP opcode 缓存。它的目标是提供一个自由、开放，和健全的框架用于缓存和优化PHP的中间代码[10]）的结果，蓝色部分是开启了APC缓存的结果。

因此，综合性能[11]、文档完善程度[12]、学习曲线三方面因素考虑，本系统选择使用了Yii框架进行开发。

4.2 MVC设计模式

MVC模式（Model-View-Controller）是软件工程中的一种软件架构模式，把软件系统分为三个基本部分：模型（Model）、视图（View）和控制器（Controller）。

MVC模式最早由Trygve Reenskaug在1978年提出[13]。MVC模式的目的是实现一种动态的程序设计，使后续对程序的修改和扩展简化，并且使程序某一部分的重复利用成为可能。除此之外，此模式通过对复杂度的简化，使程序结构更加直观:



	Yii 1.0.2	CodeIgniter 1.7.0	Zend 1.7.3	CakePHP 1.2.1	Prado 3.1.3	Symfony 1.2.2
RPS with APC	673	206	83	79	75	50
RPS without APC	93	79	38	29	24	18

图 4–1 主流PHP框架性能比较

- 控制器Controller:负责转发请求，对请求进行处理。
- 视图View:图形界面设计
- 模型Model:实现程序应有的功能（实现算法等等）、实现数据管理和数据库设计(可以实现具体的功能)。

4.3 系统MVC设计

本系统主要包含以下几个模块：

- 论文管理模块
- 科研项目管理模块
- 专利管理模块
- 人员管理模块
- 用户管理模块

以MVC模式进行设计，每个模块将包含一个模型层的模型类、一个控制器层的控制类、以及数个视图层的界面模版。又以科研项目管理模块为例，它将包含以下几个功能：

- 对外显示
- 对内显示

- 筛选
- 增加
- 删除
- 修改
- 查询

这几个功能都将分别对应科研项目控制器类中的一个动作，比如说对外显示对应动作 actionIndex ,对内显示对应动作 actionAdmin 动作,增加对应 actionCreate 动作，删除对应 actionDelete 动作等等。

如 4–2 所示，这是本系统使用MVC设计模式某一次从服务器收到客户端的HTTP请求到服务器返回给客户端HTTP响应的时序图，说明了本系统是如何运用MVC设计模式进行设计的。

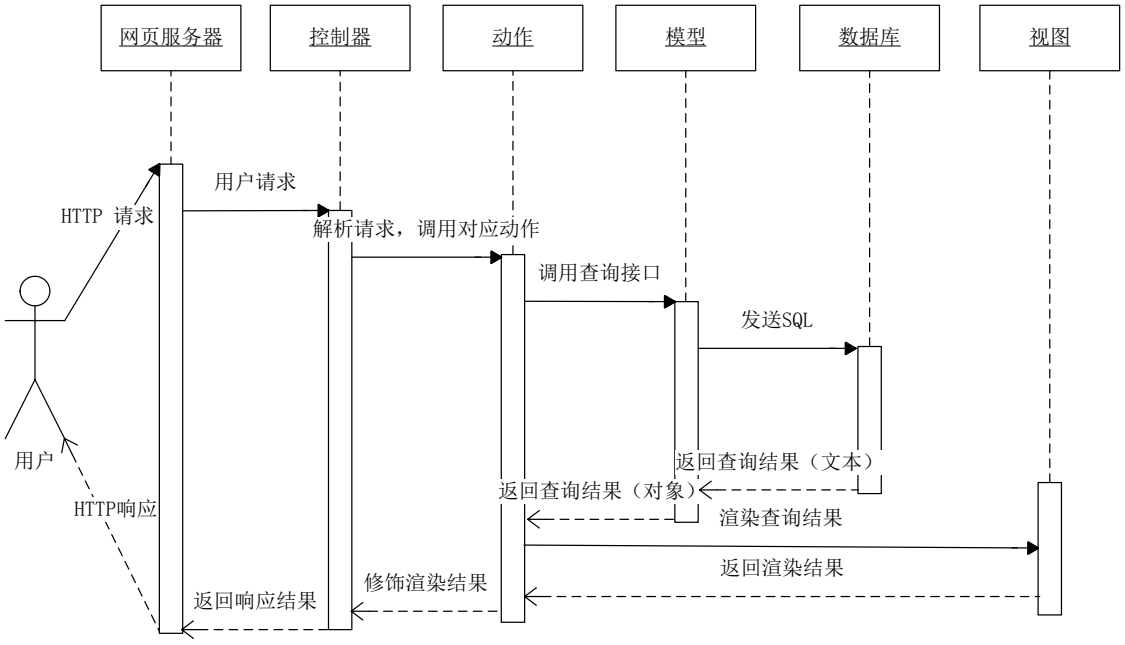


图 4–2 单次请求时序图

在这里以用户查看科研管理模块首页（对外显示）为例，分析系统的整个时序流程。

1. 用户点击“论文”超链接
2. 浏览器向URL为http://domain.com/index.php?r=project/index发送HTTP GET请求

3. Apache服务器收到请求，调用mod_php模块，对index.php执行，并且传递查询字符串（Query Parameter）：r=project/index
4. index.php解析查询字符串，调用控制器PaperController的动作actionIndex
5. 动作actionIndex调用模型Project的查询接口findAll
6. 模型Paper的查询接口findAll通过TCP Socket，发送SQL语句：“SELECT * FROM TABLE_PROJECT ”到数据库
7. 数据库返回文本查询结果，模型Paper将这些查询结果解析并使用他们例化数个新的模型类的对象返回给动作actionIndex
8. actionIndex根据这些结果，按照视图模版渲染生成HTML结果返回给控制器
9. 控制器在HTML结果的基础上进行修饰：添加头部和尾部，生成最终结果，返回给Apache
10. Apache最终结果通过HTTP响应返回给浏览器
11. 浏览器根据W3C的标准将HTTP响应绘制成响应的文字和图形，呈现给用户

4.4 总体框架设计

本系统系统主要用于科研团队的对外交流和内部管理，所以页面风格以简洁、直观为主，尽可能方便用户进行操作和使用。系统所有页面的主要框架如图 4–3 所示。本系统的所有页面都是按照图 4–3 显示的样式设计的。页面上方显示系统名称；接下来是导航栏，用户通过点击导航栏中的链接进入各个数据管理模块中去，默认进入对外展示个模块功能；导航栏右边是菜单栏，只有管理员登录过后才会显示出来，包含了对各个数据管理模块进行操作的下拉菜单，同时包括了一个登录/登出按钮。底部关于部分显示关于科研团队的简要介绍。中间的部分则显示进行各种功能操作时的信息，在执行操作和查看各种信息时，只有中间的显示部分会随着改变，显示操作结果和查看内容，而页面的顶部和底部保持不变。

4.5 数据库关系设计

本系统中需要存储的数据存在这以下的内在关系：

图 4–3 系统总体框架

- 论文与人员的关系：需要对每一篇论文条目存储一个维护人员，以及数量小于等于5的作者
- 论文与科研项目的关系：需要对每一篇论文条目存储不定量个支柱它的科研项目
- 科研项目与人员的关系：需要对每一个科研项目条目存储一个维护人员，以及不定量个实际执行人员和不定量个责任书人员
- 专利与人员的关系：需要对每一个专利项目条目存储一个维护人员，以及数量小于等于5的发明人

在这里显然不能直接在存储各项数据的数据库表中添加若干个字段，直接存储对应的信息。如表 4–1 所示，若直接在论文数据库表中加上5个字段，存储5个作者的名字，这样做会有两点坏处，一是浪费了存储空间，若只有1个作者，任然需要占用5个作者的存储空间；二是若人员数据库表有相应的变动，无法及时的反映在论文数据库表中，破坏了数据的一致性。

表 4–1 错误的存储数据间关系的做法

字段名称	数据类型	说明
id	整型	主键
info	文本	论文信息
author1	文本	第一作者的姓名
author2	文本	第二作者的姓名
author3	文本	第三作者的姓名
author4	文本	第四作者的姓名
author5	文本	第五作者的姓名

正确的做法是是要单独建立一个数据表，存储各个数据的内在关系，这样才能满足数据库第二范式，保持数据有效性，节约存储空间。如表 4–2 所示，单独建立一个论文-作者表，存储具有论文与作者的关系，应该存储论文的id与作者的id，另外还需添加一个顺序字段 seq ，用于判断是第几作者。如此一来，不仅节省了存储空间，更重要的是保证的数据的有效性和一致性。

表 4–2 论文-作者数据库表

字段名称	数据类型	说明
paper_id	整型	论文id
people_id	整型	作者id
seq	整型	取值1-5，用于判断是第几作者

本系统各个数据表之间的关系如图 4–4 所示，其中为了图标的简明性，省略了数据表的大部分字段，只给出了与数据表关系有关的字段。

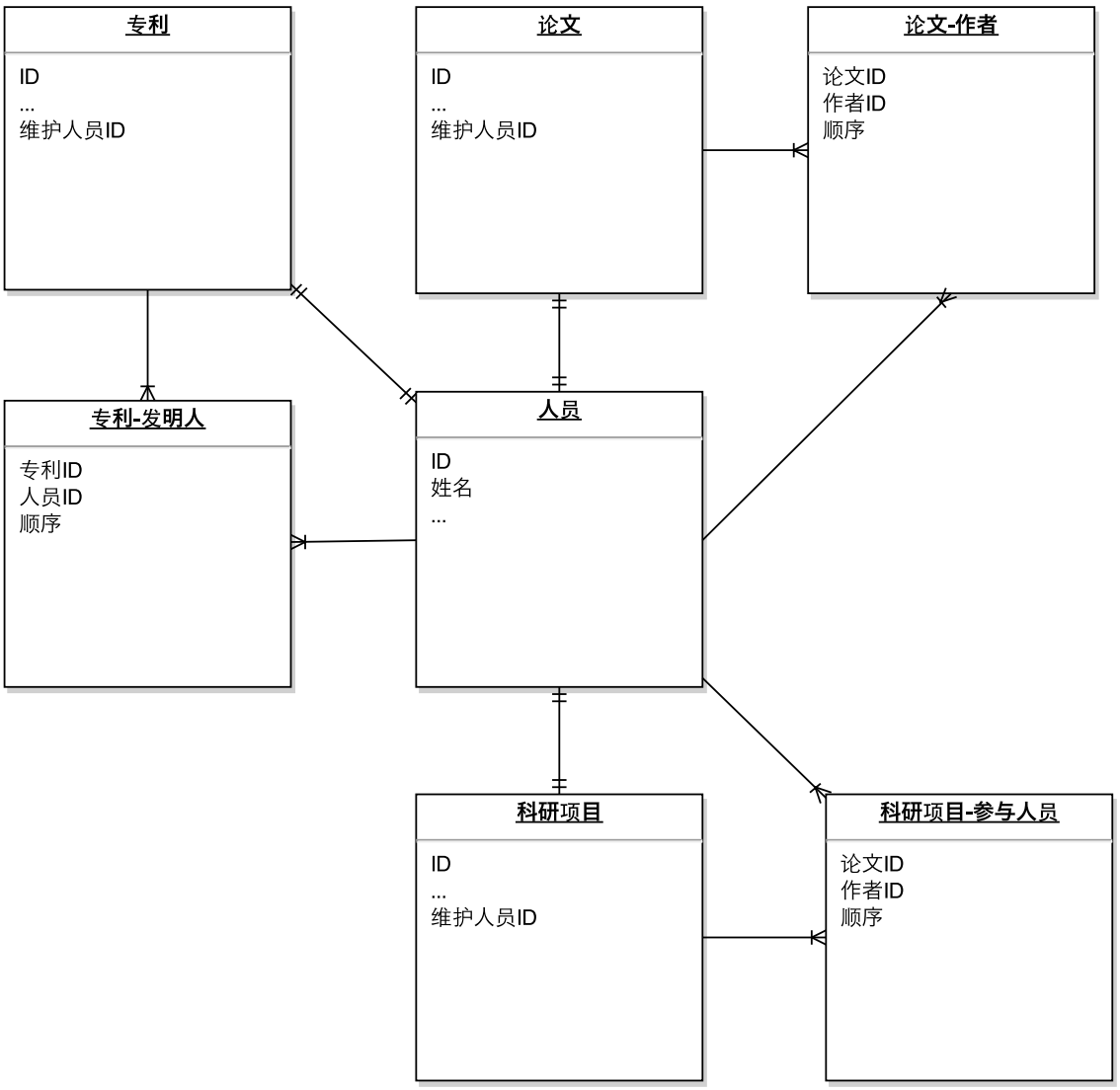


图 4–4 系统的实体-关系图

第5章 详细设计与实现

本章给出了实现本系统详细设计与实现的过程，分别给出了开发环境的搭建和配置、数据库的设计与实现、视图层的设计与实现以及模型层的设计与实现。

5.1 开发环境的搭建和配置

5.1.1 开发服务器搭建

在实现本系统之前，需要搭建出第 4 页的第 2.4 节提出的LAMP服务器环境，然后在此开发环境下进行开发与调试。首先安装 Ubuntu 12.04 LTS 操作系统，Ubuntu是一个以桌面应用为主的GNU/Linux操作系统。打开终端，输入sudo apt-get install tasksel, 安装tasksel。tasksel是一个Debian下的安装任务套件，如果你为了使你的系统完成某一种常规功能，而需要安装多个软件包时，可以使用它进行方便快捷的安装。安装成功后打开tasksel，如 5–1 所示，选择LAMP Server，一个默认配置的LAMP服务器便搭建配置完毕了。 此时，在浏览器中访问 http://localhost 就能够打开一个标题为“It works!”的默认网页，说明开发服务器已经正常运行了。

5.1.2 版本控制工具

git是一个版本控制系统，用来保留工程源代码历史状态的命令行工具。可以利用它来追踪项目中的文件，并且得到某些时间点提交的项目状态。通过使用git，可以方便地在开发本系统的过程中，任意回溯到源码的不同版本上，提高开发效率，保证了源码库的安全。图 5–2 展示了在开发本系统过程中的部分git日志，其中每个条目代表对代码库的一次提交，可以在不同的提交之间回溯。

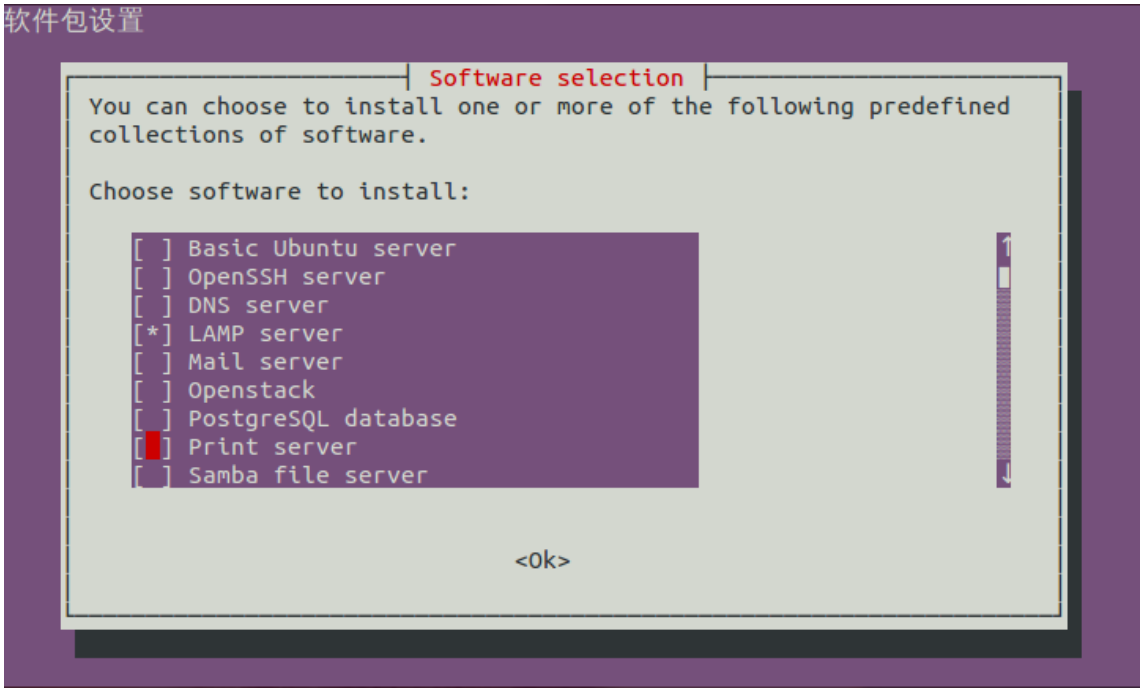


图 5–1 利用tasksel工具安装配置LAMP开发服务器

```
commit 2418a935a6a5ba928e55711e7057c548d7947e5f
Author: Zeng Luyang <zengluyang@gmail.com>
Date:   Fri May 16 12:51:45 2014 +0800

    MOD: change index page layout

commit 79d6ade8b89f6d5d1f1cf242401a85add8809a6b
Author: Zeng Luyang <zengluyang@gmail.com>
Date:   Fri May 16 12:20:49 2014 +0800

    MOD: search condition

commit ef62647eddf142adc3cefe98f343d5e31b0111fd
Author: Zeng Luyang <zengluyang@gmail.com>
Date:   Fri May 16 12:20:25 2014 +0800

    MOD: add a banner in template

commit 06f477d6b5b1a5cf3f6af93a9641ff74504fa2c2
Author: Zeng Luyang <zengluyang@gmail.com>
Date:   Fri May 16 09:39:37 2014 +0800

    MOD: implement for search options in Paper MVC
```

图 5–2 本系统开发过程中git的部分日志

5.2 数据库表实现

5.2.1 数据库引擎选择

InnoDB和MyISAM是MySQL中最常用的两个数据库引擎。MyISAM是MySQL关系数据库管理系统的默认储存引擎。这种MySQL表存储结构从旧的ISAM代码扩展出许多有用的功能。InnoDB是MySQL的另一个存储引擎，正成为目前MySQL AB所发行新版的标准，被包含在所有二进制安装包里。较之于其它的存储引擎它的优点是它支持兼容ACID的事务（类似于PostgreSQL),以及参数完整性（对外键的约束）。在新版本的MySQL中，InnoDB引擎由于其对事务，参照完整性，以及更高的并发性等优点开始广泛的取代MyISAM。

考虑到在本系统中，需要在各个数据管理模块的数据库表中保存一个维护者字段，在科研项目项目管理模块中为每个科研项目记录保存不定数量的人员，在论文管理模块中为每个项目记录保存不定数量的作者，以及在专利管理模块中为每个专利记录保存不定数量的发明人，需要使用到外键和创建额外的关系数据表，详见第页 36 第 5.2.4 节所介绍的数据表关系；考虑到InnoDB对于参数完整性的支持能够很好地保证本系统数据的一致性和完整性,本系统中的所有数据库表选用InnoDB引擎。在MySQL中指定数据库引擎非常简单，只需要在创建数据库表的SQL语句中指定“ENGINE=InnoDB”

5.2.2 数据库字符集选择

字符集是一套符号和编码的规则，不论是在Oracle数据库还是在MySQL数据库，都存在字符集的选择问题，而且如果在数据库创建阶段没有正确选择字符集，那么可能在后期需要更换字符集，而字符集的更换是代价比较高的操作，也存在一定的风险。

UTF-8（8-bit Unicode Transformation Format）是一种针对Unicode的可变长度字符编码，是用以解决国际上字符的一种多字节编码，它对英文使用8位（即一个字节），中文使用24位（三个字节）来编码。UTF-8包含全世界所有国家需要用到的字符，是国际编码，通用性强。

考虑到通用性和易用性，本系统中的所有数据库表选用UTF-8字符集。在MySQL中指定数据库字符集同样非常简单，只需要在创建数据库表的SQL语句中指定“DEFAULT CHARSET=utf8”

5.2.3 数据库表项实现

根据需求分析中提出的各数据管理模块需要实现存储的表项，在MySQL中分别创建了论文、专利、科研项目、人员四个数据库表。

5.2.3.1 论文

表 5–1 tbl_paper表的结构

字段名称	数据类型	说明
id	整型	主键，自动编号
info	文本	论文信息，不能为空
status	整形	状态
pass_date	日期	录用时间
pub_date	日期	发表时间
index_date	日期	检索时间
sci_number	变长字符串	SCI检索号
ei_number	变长字符串	EI检索号
istp_number	变长字符串	ISTP检索号
is_first_grade	布尔	是否一级
is_core	布尔	是否核心
is_journal	布尔	是否期刊
is_conference	布尔	是否会议
is_intl	布尔	是否国际
is_domestic	布尔	是否国内
file_name	变长字符串	论文文件名
file_type	变长字符串	论文文件类型
file_content	二进制数据	论文文件数据
is_high_level	布尔	是否高水平
maintainer_id	整形	维护人员id

表 5–1 给出了论文数据库表的结构，其中论文信息比较长，可能超过255个字符，因此采用MySQL中的“mediumtext”数据类型，其他变长字符串均采用“varchar(255)”数据类型，日期采用“date”数据类型，主键id与外键维护人员id采用“int(11)”数据类型，是否一级、是否核心等布尔字段均采用“tinyint(1)”型数

据类型。综合上述各字段数据类型的选择，使用下面的SQL语句，在MySQL数据库中生成论文管理模块的数据库表。

```
CREATE TABLE IF NOT EXISTS `tbl_paper` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `info` mediumtext COLLATE utf8_bin NOT NULL,
  `status` tinyint(4) DEFAULT NULL,
  `pass_date` date DEFAULT NULL,
  `pub_date` date DEFAULT NULL,
  `index_date` date DEFAULT NULL,
  `sci_number` varchar(255) COLLATE utf8_bin DEFAULT NULL,
  `ei_number` varchar(255) COLLATE utf8_bin DEFAULT NULL,
  `istp_number` varchar(255) COLLATE utf8_bin DEFAULT NULL,
  `is_first_grade` tinyint(1) DEFAULT NULL,
  `is_core` tinyint(1) DEFAULT NULL,
  `other_pub` varchar(255) COLLATE utf8_bin DEFAULT NULL,
  `is_journal` tinyint(1) DEFAULT NULL,
  `is_conference` tinyint(1) DEFAULT NULL,
  `is_intl` tinyint(1) DEFAULT NULL,
  `is_domestic` tinyint(1) DEFAULT NULL,
  `file_name` varchar(255) COLLATE utf8_bin DEFAULT NULL,
  `file_type` varchar(255) COLLATE utf8_bin NOT NULL,
  `file_size` int(11) NOT NULL,
  `file_content` mediumblob NOT NULL,
  `is_high_level` tinyint(1) DEFAULT NULL,
  `maintainer_id` int(11) DEFAULT NULL,
  PRIMARY KEY (`id`),
  KEY `tbl_paper_ibfk_1` (`maintainer_id`)
) ENGINE=InnoDB  DEFAULT CHARSET=utf8 COLLATE=utf8_bin;
```

5.2.3.2 科研项目

表 5–2 tbl_project表的结构

字段名称	数据类型	说明
id	整型	主键，自动编号
name	变长字符串	项目名称
number	变长字符串	编号
fund_number	变长字符串	经费本编号
is_intl	布尔	是否国际
is_national	布尔	是否国家级
is_provincial	布尔	是否省部级
is_city	布尔	是否市级
is_school	布尔	是否校级
is_enterprise	布尔	是否横向
is_NSF	布尔	是否国家自然基金
is_973	布尔	973
is_863	布尔	863
is_NKTRD	布尔	是否科技支撑计划
is_DFME	布尔	是否教育部博士点专项基金
is_major	布尔	是否重大专项
start_date	日期	开始时间
deadline_date	日期	截至时间
conclude_date	日期	结题时间
app_date	日期	申报时间
pass_date	日期	立项时间
app_fund	货币	申报经费
pass_fund	货币	立项经费

表 5–2 给出了科研项目数据库表的结构，其中所有类型为变长字符串均的字段均采用MySQL中的“varchar(255)”数据类型，日期采用“date”数据类型，主键id与外键维护人员id采用“int(11)”数据类型，是否国际、是否国家级等类型为布尔的字段均采用“tinyint(1)”型数据类型；申报经费、立项经费不使用字符串类型或浮点类型存储，而是采用整数部分为15位，小数部分为2位的数值类型存储，方便比较和计算，且没有误差，在MySQL对应“decimal(15,2)”数据类型。综合上述各字段数据类型的选择，使用下面的SQL语句，在MySQL数据库中生成

科研项目管理模块的数据库表。

```
CREATE TABLE IF NOT EXISTS `tbl_project` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `name` varchar(255) COLLATE utf8_bin DEFAULT NULL,
  `number` varchar(255) COLLATE utf8_bin DEFAULT NULL,
  `fund_number` varchar(255) COLLATE utf8_bin DEFAULT NULL,
  `is_intl` tinyint(1) DEFAULT NULL,
  `is_national` tinyint(1) DEFAULT NULL,
  `is_provincial` tinyint(1) DEFAULT NULL,
  `is_city` tinyint(1) DEFAULT NULL,
  `is_school` tinyint(1) DEFAULT NULL,
  `is_enterprise` tinyint(1) DEFAULT NULL,
  `is_NSF` tinyint(1) DEFAULT NULL,
  `is_973` tinyint(1) DEFAULT NULL,
  `is_863` tinyint(1) DEFAULT NULL,
  `is_NKTRD` tinyint(1) DEFAULT NULL,
  `is_DFME` tinyint(1) DEFAULT NULL,
  `is_major` tinyint(1) DEFAULT NULL,
  `start_date` date DEFAULT NULL,
  `deadline_date` date DEFAULT NULL,
  `conclude_date` date DEFAULT NULL,
  `app_date` date DEFAULT NULL,
  `pass_date` date DEFAULT NULL,
  `app_fund` decimal(15,2) DEFAULT NULL,
  `pass_fund` decimal(15,2) DEFAULT NULL,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB  DEFAULT CHARSET=utf8 COLLATE=utf8_bin;
```

5.2.3.3 专利

表 5–3 tbl_people表的结构

字段名称	数据类型	说明
id	整形	主键，自动编号
name	变长字符串	专利名称
app_date	日期	申请时间
app_number	变长字符串	申请号
auth_date	日期	授权时间
auth_number	变长字符串	授权号
is_intl	布尔	是否国际
is_domestic	布尔	是否国内
abstract	变长字符串	专利摘要

表 5–4 给出了专利数据库表的结构，其中所有类型为变长字符串均的字段均采用MySQL中的“varchar(255)”数据类型，日期采用“date”数据类型，主键id“int(11)”数据类型，是否国际、是否国家级等类型为布尔的字段均采用“tinyint(1)”型数据类型；申报经费、立项经费不使用字符串类型或浮点类型存储，而是采用整数部分为15位，小数部分为2位的数值类型存储，方便比较和计算，且没有误差，在MySQL对应“decimal(15,2)”数据类型。综合上述各字段数据类型的选择，使用下面的SQL语句，在MySQL数据库中生成专利模块的数据库表。

```
CREATE TABLE IF NOT EXISTS `tbl_patent` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `name` varchar(255) COLLATE utf8_bin NOT NULL,
  `app_date` date NOT NULL,
  `app_number` varchar(255) COLLATE utf8_bin NOT NULL,
  `auth_number` varchar(255) COLLATE utf8_bin DEFAULT NULL,
  `auth_date` date DEFAULT NULL,
  `is_intl` tinyint(1) NOT NULL,
  `is_domestic` tinyint(1) NOT NULL,
  `abstract` text COLLATE utf8_bin NOT NULL,
  PRIMARY KEY (`id`)
```

```
) ENGINE=InnoDB  DEFAULT CHARSET=utf8 COLLATE=utf8_bin;
```

5.2.3.4 人员

表 5–4 tbl_people表的结构

字段名称	数据类型	说明
id	整形	主键，自动编号
name	变长字符串	姓名
name_zh	日期	姓名拼音或英文名
type	布尔	教师或者学生
description	变长字符串	介绍

表 5–4 给出了人员数据库表的结构，其中所有类型为变长字符串均的字段均采用MySQL中的“varchar(255)”数据类型,主键id“int(11)”数据类型，type字段采用“tinyint(1)”型数据类型；综合上述各字段数据类型的选择，使用下面的SQL语句，在MySQL数据库中生成人员模块的数据库表。

```
CREATE TABLE IF NOT EXISTS ‘tbl_people‘ (
  ‘id‘ int(11) NOT NULL AUTO_INCREMENT,
  ‘name‘ varchar(255) COLLATE utf8_bin NOT NULL,
  ‘name‘ varchar(255) COLLATE utf8_bin,
  ‘type‘ tinyint(1),
  ‘description‘ varchar(255),
  PRIMARY KEY (‘id‘)
) ENGINE=InnoDB  DEFAULT CHARSET=utf8 COLLATE=utf8_bin;
```

5.2.3.5 用户

表 5–5 tbl_user表的结构

字段名称	数据类型	说明
id	整形	主键，自动编号
username	变长字符串	用户名
password	变长字符串	进行散列运算后的密码
email	变长字符串	电子邮箱地址
description	变长字符串	介绍
is_admin	布尔	是否为超级管理员
is_paper	布尔	是否为论文模块管理员
is_project	布尔	是否为科研项目模块管理员
is_patent	布尔	是否为专利模块管理员

表 5–5 给出了人员数据库表的结构，其中所有类型为变长字符串均的字段均采用MySQL中的“varchar(255)”数据类型,主键id使用“int(11)”数据类型，类型为布尔的字段采用“tinyint(1)”数据类型；综合上述各字段数据类型的选择，使用下面的SQL语句，在MySQL数据库中生成人员模块的数据库表。

```
CREATE TABLE IF NOT EXISTS `tbl_user` (
  `id` int(11) NOT NULL AUTO_INCREMENT,
  `username` varchar(30) COLLATE utf8_bin NOT NULL,
  `password` varchar(255) COLLATE utf8_bin NOT NULL,
  `email` varchar(100) COLLATE utf8_bin DEFAULT NULL,
  `is_admin` tinyint(1) DEFAULT NULL,
  `is_paper` tinyint(1) DEFAULT NULL,
  `is_project` tinyint(1) DEFAULT NULL,
  `is_patent` tinyint(1) DEFAULT NULL,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB  DEFAULT CHARSET=utf8 COLLATE=utf8_bin;
```

5.2.4 数据表关系

在科研项目管理中，需要存储和维护不定数目的项目参与人员，科研项目和

参与人员之间存在着多对多的关系。在这里显然不能直接在科研项目的数据表中添加若干个字段，存储人员的id；而是要单独建立一个数据表，存储项目和参与人员之间的关系，这样才能满足数据库第二范式，保持数据有效性，节约存储空间。

所谓数据库第二范式（2NF），它要求实体的属性完全依赖于主关键字。所谓完全依赖是指不能存在仅依赖主关键字一部分的属性，如果存在，那么这个属性和主关键字的这一部分应该分离出来形成一个新的实体，新实体与原实体之间是一对多的关系。为实现区分通常需要为表加上一个列，以存储各个实例的惟一标识。简而言之，第二范式就是属性完全依赖于主键。

表 5–6 给出了存储科研项目与实际执行人员的关系的数据表结构。这样在查找某一个科研项目记录的时候，只需要在存储科研项目与实际执行人员的关系的数据表中找到满足科研项目id等于当前科研项目记录id，就可以得到该项目的所有实际执行人员id（可能有多个），再依次地按照这些际执行人员id在人员数据表中查找，便可以得到某一个科研项目的所有实际执行人员。

表 5–6 tbl_project_people_execute表的结构

字段名称	数据类型	说明
project_id	整形	科研项目id，外键，不能为空
people_id	整形	人员id，外键，不能为空
seq	整形	人员在单个项目中的顺序

使用下面的SQL语句生成存储科研项目与实际执行人员的关系的数据表:

```
CREATE TABLE IF NOT EXISTS `tbl_project_people_execute` (
  `project_id` int(11) NOT NULL,
  `people_id` int(11) NOT NULL,
  `seq` int(11) NOT NULL,
  PRIMARY KEY (`project_id`,`people_id`),
) ENGINE=InnoDB DEFAULT CHARSET=utf8 COLLATE=utf8_bin;
```

使用下面的SQL语句给该数据表添加外键限制，以增强本系统数据的有效性和一致性：

```
ALTER TABLE `tbl_project_people_execute`
  ADD CONSTRAINT `tbl_project_people_execute_ibfk_1`
```

```
    FOREIGN KEY (`project_id`)
    REFERENCES `tbl_project` (`id`) ON DELETE CASCADE ON ↩
    UPDATE CASCADE,
  ADD CONSTRAINT `tbl_project_people_execute_ibfk_2`
    FOREIGN KEY (`people_id`)
    REFERENCES `tbl_people` (`id`) ON DELETE CASCADE ON ↩
    UPDATE CASCADE;
```

通过使用SQL的“JOIN”命令，可以从存储科研项目与实际执行人员的关系的数据表中得到某个项目的实际执行人员。例如，下面的SQL查询得到id为1的科研项目实际执行人员：

```
SELECT `execute_`.`id` AS `t1_c0`, `execute_`.`name` AS
`t1_c1` FROM `tbl_people` `execute_`  INNER JOIN
`tbl_project_people_execute` `execute_peoples_execute_` ON
(`execute_peoples_execute_`.`project_id`=1) AND
(`execute_`.`id`=`execute_peoples_execute_`.`people_id`) ↩
ORDER BY
execute_peoples_execute_.seq;
```

如图 5–3 所示，这是与科研项目有关的实体-关系图（ER图），图中展示了科研项目与实际执行人员的关系、科研项目与责任书人员的关系、论文与支柱科研项目的关系以及论文与报账科研项目之间的关系。

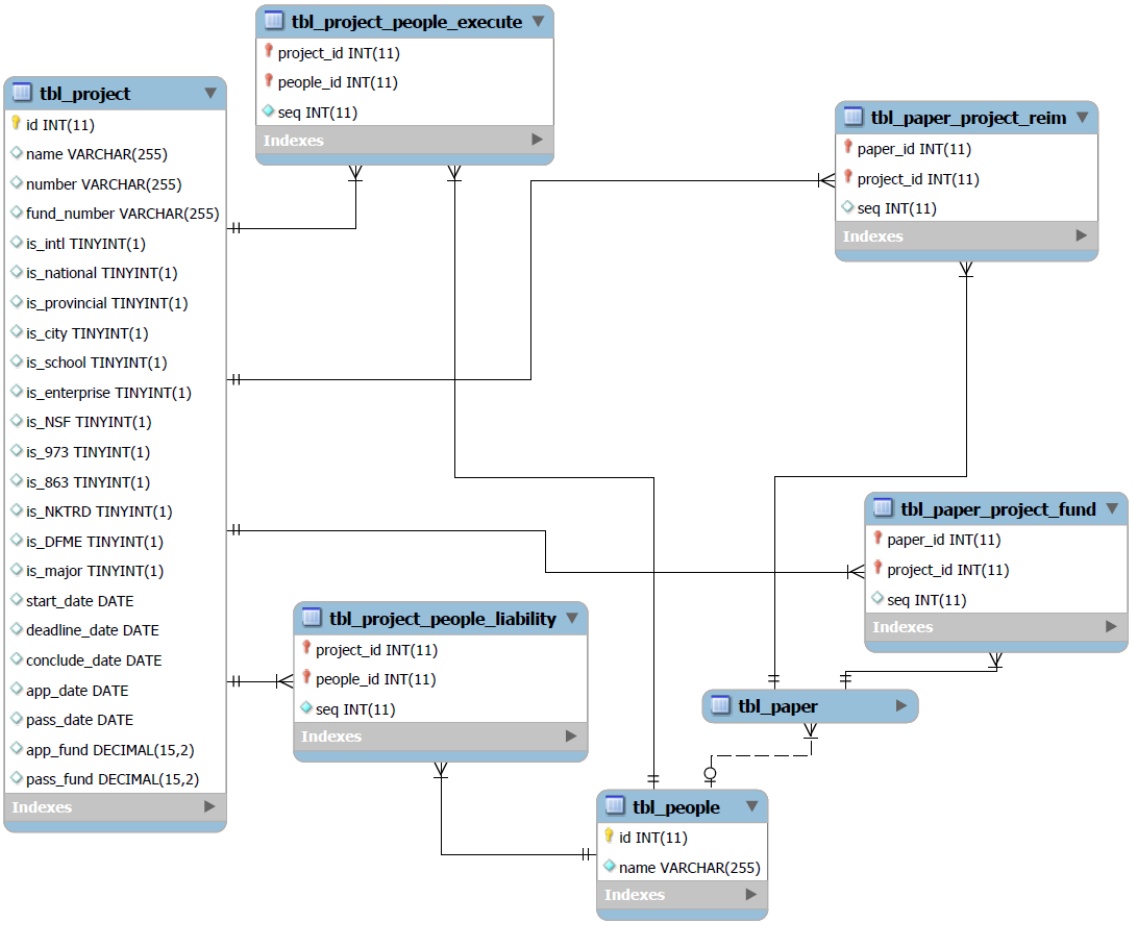


图 5–3 与科研项目有关的实体-关系图

同样地，在本系统中，还需要分别存储和维护科研项目与责任书人员的关系、论文与支柱科研项目的关系、论文与报账科研项目之间的关系、科研项目与维护人员的关系、论文与作者的关系、论文与维护人员的关系、发明和发明人的关系，皆采用类似上述建立一个关系数据库表的方式来实现，限于篇幅关系，在这里不再赘述。

5.3 MVC开发规范

在 Yii 框架中，偏爱规范胜于配置。遵循规范可使你能够创建成熟的Yii应用而不需要编写、维护复杂的配置。

下面我们讲解Yii 编程中推荐的开发规范。为简单起见，我们假设WebRoot是Yii 应用安装的目录。

5.3.1 URL

默认情况下，Yii 识别如下格式的URL：

http://hostname/index.php?r=ControllerID/ActionID

其中， r 是一个GET的查询参数，意为路由（route），它可以被Yii解析为控制器和动作。如果ActionID 被省略，控制器将使用默认的动作（在CController::defaultAction中定义）；如果ControllerID 也被省略（或者r 变量不存在），应用将使用默认的控制器（在CWebApplication::defaultController中定义）。

5.3.2 文件

命名和使用文件的规范取决于它们的类型。

类文件应以它们包含的公有类命名。例如，CController 类位于CController.php文件中。公有类是可以被任何其他类使用的类。每个类文件应包含最多一个公有类。私有类（只能被一个公有类使用的类）可以放在使用此类的公有类所在的文件中。

视图文件应以视图的名字命名。例如，index 视图位于index.php 文件中。视图文件是一个PHP脚本文件，它包含了用于呈现内容的HTML和PHP代码。

配置文件可以任意命名。配置文件是一个PHP脚本，它的主要目的是返回一个体现配置的关联数组。

5.3.3 目录

Yii 假定了一系列默认的目录用于不同的场合：

- WebRoot/protected: 这是应用基础目录，是放置所有安全敏感的PHP脚本和数据文件的地方。Yii 有一个默认的application 别名指向此目录。此目录及目录中的文件应该保护起来防止Web用户访问。
- WebRoot/protected/models:此目录放置所有的模型文件。
- WebRoot/protected/controllers: 此目录放置所有控制器类文件。
- WebRoot/protected/views: 此目录放置所有试图文件，包含控制器视图，布局视图和系统视图。它可以通过
- WebRoot/protected/views/ControllerID: 此目录放置单个控制器类中使用的视图文件。此处的ControllerID 是指控制器的ID 。

5.4 模型层实现

在本小节中，仅给出了数据表项最多，数据关系最复杂的科研项目模型的实现，人员模型、论文模型、专利模型的实现方法与科研项目模型相类似，在这里不再赘述。

Yii框架提供了易用的AR(Active Record)实现，AR 是一个流行的对象-关系映射(ORM) 技术。每个AR 类代表一个数据表（或视图），数据表（或视图）的列在AR 类中体现为类的属性，一个AR 实例则表示表中的一行。常见的CRUD 操作作为AR 的方法实现。在本系统中，使用AR技术来实现各个模型。

按照第 5.3.3 节中所提高到的文件与目录规范，应该将本小节中所提到的科研项目模型类的实现，放在 WebRoot/protected/models/ 目录中的 Project.php 文件中。

5.4.1 建立数据库连接

AR 依靠一个数据库连接以执行数据库相关的操作。在Yii框架目录下的config.php中按照下面的PHP代码配置数据库链接。其中，host是MySQL服务器的地址，在开发服务器上，即为本机，填写本地环回地址；dbname为MySQL中存储本系统的数据库名称，根据需要填写，在开发服务器上填写为testdrive；username和password分别为mysql的密码；charset填写在第 29页 第 5.2.2 节中选择的utf-8。

```
return array(
    //其它配置
    'db'=>array(
            'connectionString' => 'mysql:host=127.0.0.1;
            dbname=testdrive',
            'emulatePrepare' => true,
            'username' => 'root',
            'password' => 'test',
            'charset' => 'utf8',
        ),
);
```

5.4.2 定义AR类

Yii 框架提供了一个名为 CActiveRecord 的 AR 类，如下面的 PHP 代码所示：让科研项目模型类 Project 继承自 CActiveRecord 类，并复写父类的 tableName() 方法，使它返回我们存储科研项目的数据库表的名称，即“tbl_project”，便可以使用 CActiveRecord 提供的接口以面向对象的方式对科研项目的数据库表进行增加、删除、修改、查询了。

```
class Project extends CActiveRecord
{
    public function tableName()
    {
        return 'tbl_project';
    }
}
```

5.4.3 实现增删操作

增删改查操作指对数据库表记录的增加、删除、修改和查询，在本系统的控制器层中，可以使用面向对象的方式对科研项目的数据库表进行操作了：

1. 增加新的科研项目的条目：

```
$project=new Project;
$project->name='项目名称';
//对其他字段进行赋值
$project->save();
```

2. 对现有科研项目的条目进行查询：

```
// 查找满足指定条件的结果中的第一行
$project=Project::model()->find($condition,$params);
// 查找具有指定主键值的那一行
$project=Project::model()->findByPk($projectID,$condition,
$params);
// 查找具有指定属性值的行
```

```
$project=Project::model()->findByAttributes($attributes,
$condition,$params);
// 通过指定的 SQL 语句查找结果中的第一行
$project=Project::model()->findBySql($sql,$params);

$projects=Project::model()->findAll($condition,$params);
// 查找带有指定主键的所有行
$projects=Project::model()->findAllByPk($projectIDs,
$condition,$params);
// 查找带有指定属性值的所有行
$projects=Project::model()->findAllByAttributes($attributes,
$condition,$params);
// 通过指定的SQL语句查找所有行
$projects=Project::model()->findAllBySql($sql,$params);
```

3. 对现有科研项目的条目进行修改：

```
$project=Project::model()->findByPk(10);
$project->name='新的项目名称';
$project->save(); // 将更改保存到数据库
```

4. 对现有科研项目的条目进行删除：

```
$project=Project::model()->findByPk(10); // 
假设有一个科研项目，其 ID 为 10
$project->delete(); // 从数据表中删除此行

// 删除符合指定条件的行
Project::model()->deleteAll($condition,$params);
// 删除符合指定条件和主键的行
Project::model()->deleteByPk($pk,$condition,$params);
```

5.4.4 定义数据库表关系

在上一节中，已经实现了使用Active Record (AR) 从单个数据表对条目进行增删改查的操作。在本节中，将使用AR 连接多个相关数据表并取回关联（对应MySQL中的“JOIN”语句）后的数据表。

在使用AR 执行关联查询之前，需要让AR 知道一个AR 类是怎样关联到另一个的。

两个AR 类之间的关系直接通过AR 类所代表的数据表之间的关系相关联。在AR 中，定义了三种关系类型：

1. BELONGS_TO：一对多
2. HAS_MANY：多对一
3. MANY_MANY:多对多

在本系统的科研项目管理模块中，科研项目与维护者之间的关系是多对一，而科研项目与执行人员之间的关系是多对多。

如下面的PHP代码所示，通过在Project类中复写父类 AR 中定义关系的 relations() 方法,可以定义科研项目与其它数据库表的关系。

```
public function relations()
{
    return array(
        'liability_peoples' => array(
            self::MANY_MANY,
            'People',
            'tbl_project_people_liability(project_id, ↩
            people_id)',
            'order'=>'liability_peoples_liability_.seq',
            'alias'=>'liability_'
        ),
        'execute_peoples' => array(
            self::MANY_MANY,
            'People',
            'tbl_project_people_execute(project_id, ↩
            people_id)',
```

```
            'order'=>'execute_peoples_execute_.seq',
            'alias'=>'execute_'
        ),
    );

}
```

在定义了数据库表关系之后，可以在控制器层中方便地执行关联查询以及增删改查，就像访问科研项目的模型类Project本身的属性一样：

```
//获取 ID 为 10 的科研项目
$project=Project::model()->findByPk(10);
//获取此科研项目的维护者的姓名：)
此处将执行一个关联查询。
$maintainter=$project->maintainter->name;
//获取此科研项目的所有实际执行人员
//将返回一个人员对象的数组
$exec_peoples=$project->execute_peoples;
```

5.4.5 验证数据有效性

当插入或更新一行时，我们常常需要检查列的值是否符合相应的规则。如果列的值是由最终用户提供的，这一点就更加重要。总体来说，为了防止恶意攻击，永远不能相信任何来自客户端的数据。为了维护本系统数据的有效性，在本系统的模型层需要定义验证输入数据有效性的规则。

当调用 AR 类的实例的save()方法时，AR 类的实例会自动执行数据验证。验证是基于在 AR 类的rules() 方法中指定的规则进行的。如下面给出的PHP代码，通过根据在第 32 页第 5.2.3.2 节建立的数据库表的结构，在科研项目模型类Project中复写父类的rules()方法，可以完成对数据验证规则的定义。这样以来，控制层便能判断用户输入、提交数据是否有效，避免了无效数据的录入或者是恶意攻击的可能性。

```
public function rules()
{
    return array(
        array('name','required'),
        array('is_intl, is_national, is_provincial, is_city, ↩
        is_school, is_enterprise, is_NSF, is_973, is_863, ↩
        is_NKTRD, is_DFME, is_major', 'numerical', ↩
        'integerOnly'=>true),
        array('name, number, fund_number', 'length', 'max'=>↩
        255),
        array('app_fund, pass_fund', 'length', 'max'=>15),
        array('start_date, deadline_date, conclude_date, ↩
        app_date, pass_date', 'safe'),
        array('id, name, number, fund_number, is_intl, ↩
        is_national, is_provincial, is_city, is_school, ↩
        is_enterprise, is_NSF, is_973, is_863, is_NKTRD, ↩
        is_DFME, is_major, start_date, deadline_date, ↩
        conclude_date, app_date, pass_date, app_fund, ↩
        pass_fund, ', 'safe', 'on'=>'search'),
    );
}
```

5.4.6 搜索与筛选

Yii框架中的 CDbCriteria 类代表了一条数据库查询的条件，比如说 MySQL 中的 WHERE 字句、 AND 运算符、 OR 运算符、 ORDER BY 子句等。利用它来创建在搜索与查询中需要添加的条件，生成对应的SQL语句。利用 CDbCriteria 类提供的compare()方法，可以实现搜索与筛选功能。compare()的作用是添加一个比较条件到最终生成的SQL语句的 WHERE 字句中去。还可以通过设置 CDbCriteria 类的 condition 、 group 、 order 等属性，配置SQL中对应的条件。

CActiveDataProvider 类使用AR的CActiveRecord::findAll()方法，从数据库中检索信息。它的criteria属性是 CDbCriteria 类的一个实例，能够用来查询多种指定条件。

利用 CDbCriteria 类，实现了科研项目模型类 Project 中的搜索方法 search()，提供给本系统的控制器层调用。search() 方法将 Project 类的各个属性通过一个 CDbCriteria 实例的 compare() 方法添加到了查询条件中，最后利用这些查询条件作为一个 CActiveDataProvider 类的实例构造函数的参数，传递给这个 CActiveDataProvider 类的实例，并返回给供模型层使用。

5.5 视图层实现

视图层的功能主要是定义一个科研项目数据输入、显示的格式与模版，供控制器层调用。

5.5.1 整体布局

本系统视图层的所有页面的主体页面布局形式如图 5–4 所示，其中页面顶部为系统的LOGO以及名称；然后下面是导航栏，导航栏左侧部分显示到系统各个模块的链接，右侧部分显示当前模块当前登录用户名称、可供操作的菜单选项以及登录、登出按钮；然后底部的是footer部分，显示关于本系统的总体介绍；中间部分则为各模块各个功能显示的内容。前三部分（顶部LOGO、导航栏、底部footer）在本系统的每个页面都是相同的，提取出来作为本系统的主题模版，而每次渲染的时候只需要渲染中间部分的内容即可，这样可以提高运行效率和系统样式的一致性。

根据在第 14 页第 3.3.2 节所需要实现的科研项目录入功能，在本系统的视图层实现了分别实现用于批量录入和逐个录入的界面。批量录入与逐个录入界面的实现分别放在 WebRoot/protected/views/projects 目录中的 upload.php 和 create.php 中

图 5–4 系统整体布局

按照第 5.3.3 节中所提高到的文件与目录规范，应该将本小节中所提到的科研项目视图层的实现，放在 WebRoot/protected/views/projects 目录中，每个界面单独放在该目录的一个文件中。

5.5.2 录入界面

批量录入科研项目的界面比较简单，只需要提供一个提示用户选择所需要批量录入的Excel文件，以及一个用于确认上传的按钮即可，如图 5–5 所示。

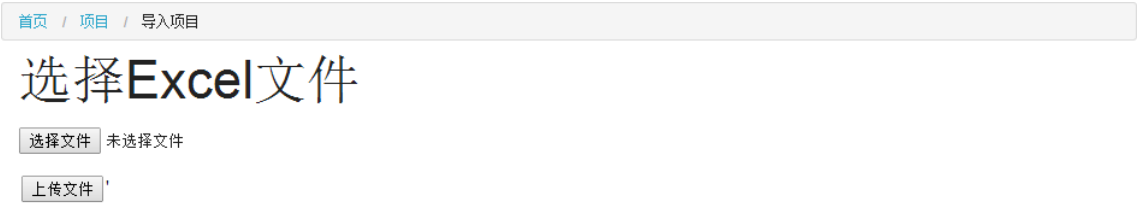


图 5–5 批量录入界面

如图 5–6 所示逐个录入的界面分别按照录入内容的要求实现了文本输入框、下拉菜单等输入表单，鉴于表单较多，响应的式布局技术①，在较大屏幕上实现多栏显示，同一行可以显示多个输入控件，提高屏幕的利用率；而在较小的屏幕，如手机的屏幕上，自动地调制为单栏显示，提高本系统在较小屏幕下的使用体验。

(a) (b)

图 5–6 逐个录入界面
(a)较大屏幕上多栏显示；(b)较小屏幕上自适应为单栏显示

另外指的提到的是，在逐个录入科研项目的过程中，用户需要选择科研项目的参与人员，此时如果仅仅显示一个下拉菜单供用户进行人员的选择，由于人员较多，选择起来不方便容易出错。因此，如图 5–7 所示，在录入界面人员输入窗口，实现了一个输入与下拉菜单相结合的输入控件，用户可以直接进行选择，也可通过拼音首字母，缩小选择的范围，快速、方便、人性化地进行录入。

5.5.3 修改界面

修改界面实际上和根据在第 48 页第 5.5.2 节所实现的录入界面是一样，只是在相应的输入控件中，显示当前修改科研项目条目已有的内容，如图5–8 所示。修改界面的实现分别放在 WebRoot/protected/views/projects 目录中的 update.php 中

① 响应式布局是Ethan Marcotte在2010年5月份提出的一个概念，简而言之，就是一个网站能够兼容多个终端——而不是为每个终端做一个特定的版本。这个概念是为解决移动互联网浏览而诞生的。

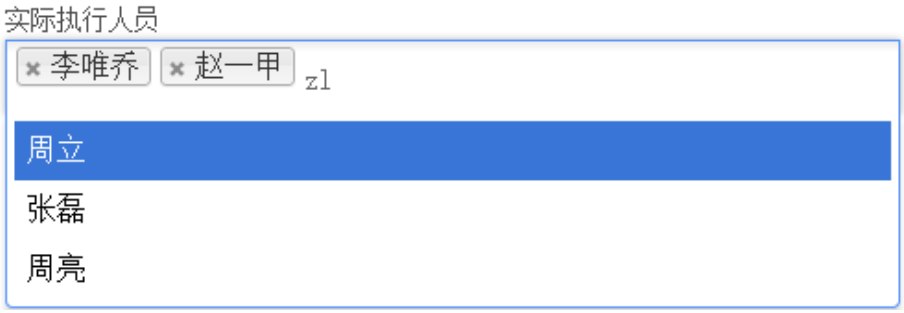


图 5–7 人性化的录入界面

图 5–8 修改界面

5.5.4 管理界面

分析第 15 页第 3.3.4 节中提出的修改与删除功能，第 15 页第 3.3.5 节中提出的查询功能可以发现，通常在用户的使用本系统过程中，是按照 5–9 所示的流程来使用上述功能的，也就是说修改、删除、查询三个功能经常需要一起使用，所以在这里将这三个功能集中到一个管理页面上，如图 5–9 所示。用户通过点击“筛选与查找按钮”，可以弹出如图 5–10 所示筛选表单，供用户选择，从而查询，筛选出目标项目；然后点击对应的按钮，查看其详细内容、进行修改、进行删除。管理界面的实现分别放在 WebRoot/protected/views/projects 目录中的 admin.php 中

图 5–9 管理界面

图 5–10 筛选界面

5.5.5 显示界面

对外显示：如图 5–11 所示，采用表格的形式对角色为游客的用户进行显示。对外显示界面的实现分别放在 WebRoot/protected/views/projects 目录中的 index.php 中

对内显示：如图 5–11 ，即通过上一节中所实现的管理页面对角色为超级管理员、科研项目模块管理员的用户进行显示，在每一条科研项目条目，提供三个按钮，分别对应查看其详细内容、进行修改、进行删除，如图 5–12 所示。对内显示界面的实现分别放在 WebRoot/protected/views/projects 目录中的 admin.php 中

5.6 控制器层实现

在在科研项目控制层中，主要完成调用科研项目模型层提供的增加、删除、修改、查询接口，获取对应的数据，将它们按照视图层中的模版显示出来，以实现在第 13 页第 3.3 节需求分析中提出的各项功能。

首页 / 科研项目

科研项目

共 20 条.

项目级别类型	项目名称	时间段
国家级，国家自然基金	基于多信道分配的认知网络信道分配策略研究	2011年01月~2013年12月
国家级，国家自然基金	车用无线自组织网络安全告警信息传输策略研究	2009年01月~2011年12月
国家级，国家自然基金	基于认知无线电的通信抗干扰理论与技术	2009年01月~2012年12月
国家级，863	混合式动态频谱资源共享宽带无线通信协议的设计与实现	2009年04月~2011年06月
国家级，重大专项	超高速无线局域网无线接口关键技术研究与验证	2010年01月~2012年12月
国家级，重大专项	超高速无线局域网无线接口关键技术研究与验证	2010年01月~2012年12月
国家级，重大专项	宽带无线校园创新实验网体系架构与关键技术研究	2010年01月~2012年06月
国家级，重大专项	无线局域网与蜂窝移动通信网络融合技术研究与验证	2010年01月~2011年12月
国家级，重大专项	全IP宽带移动网络架构及关键技术研究	2009年01月~2010年12月
国家级，重大专项	传感器网络总体研究及仿真平台	2008年11月~2010年12月
省部级	新一代宽带无线网络信道资源管理技术研究	2011年01月~2013年12月
省部级	新世纪优秀人才支持计划项目	2011年01月~2013年12月
省部级	XX抗干扰模型研究	2009年05月~2011年04月
省部级，教育部高校博士点基金	车用无线自织网交通安全告警信息传输策略研究	2009年01月~2011年12月
省部级，教育部高校博士点基金	基于分层结构的无线Mesh网络关键技术研究	2008年08月~2010年06月
横向	基于IPv6的工业WSN组网与节能MAC技术	2010年04月~2010年10月
横向	先进传感器网络技术的研究	2009年12月~2011年12月
国家级，科技支撑计划	龙门山地震带小流域滑坡泥石流灾害监测预警技术研究与示范	2011年04月~2015年03月
国家级，科技支撑计划	滑坡泥石流灾害监测预警技术集成与应用示范	2010年04月~2015年03月
国家级	跳频体制下Ad Hoc组网技术研究	2008年01月~2009年12月

图 5–11 对外显示界面

图 5–12 对内显示界面

按照第 5.3.3 节中所提高到的文件与目录规范，应该将本小节中所提到的科研项目控制器类的实现，放在 WebRoot/protected/controllers/ 目录中的 ProjectController.php 文件中。

5.6.1 录入

考虑到数据录入过程繁琐性，本系统提供了批量录入功能，用户可以上传一个包含科研项目数据的Excel表格，一次性的录入多个数据，方便了用户的使用。又考虑到后续使用过程中，单个增加项目的需求，本系统同时实现了单个录入功能，能够逐个地向系统添加数据。

5.6.1.1 批量录入

当用户点击“导入科研项目”时，会最终调用到控制器 ProjectController 的动作 actionImport，在这个动作中实现了逐个录入功能。程序的流程图如图 5–13 所示。

在实现批量录入功能时使用到的一个名为 PHPExcel 的第三方库，它的功能是将 Microsoft Excel 文件转换为一个PHP的二维数组。通过使用 PHPExcel 的这个功能，在本系统中，遍历这个数组的每一行，对于每一行数据，新例化一个科研项目模型类，对相应的属性进行赋值，最后调用模型类的Save方法，便可以将Excel中的科研项目的数据逐个存储到数据库中去。另外，同样需要存储科研项

目的参与人员，即科研项目与人员之间的关系。此时，在处理那个二维数组每一行的过程中，遇到需要存储的人员关系，首先需要判断那个人员是否已存储在人员数据库中，若没有，则需要新建一个对应的人员。然后再在科研项目与人员关系数据表中，即:tbl_project_people_excecute 和tbl_project_people_liability ，存储对应的关系。

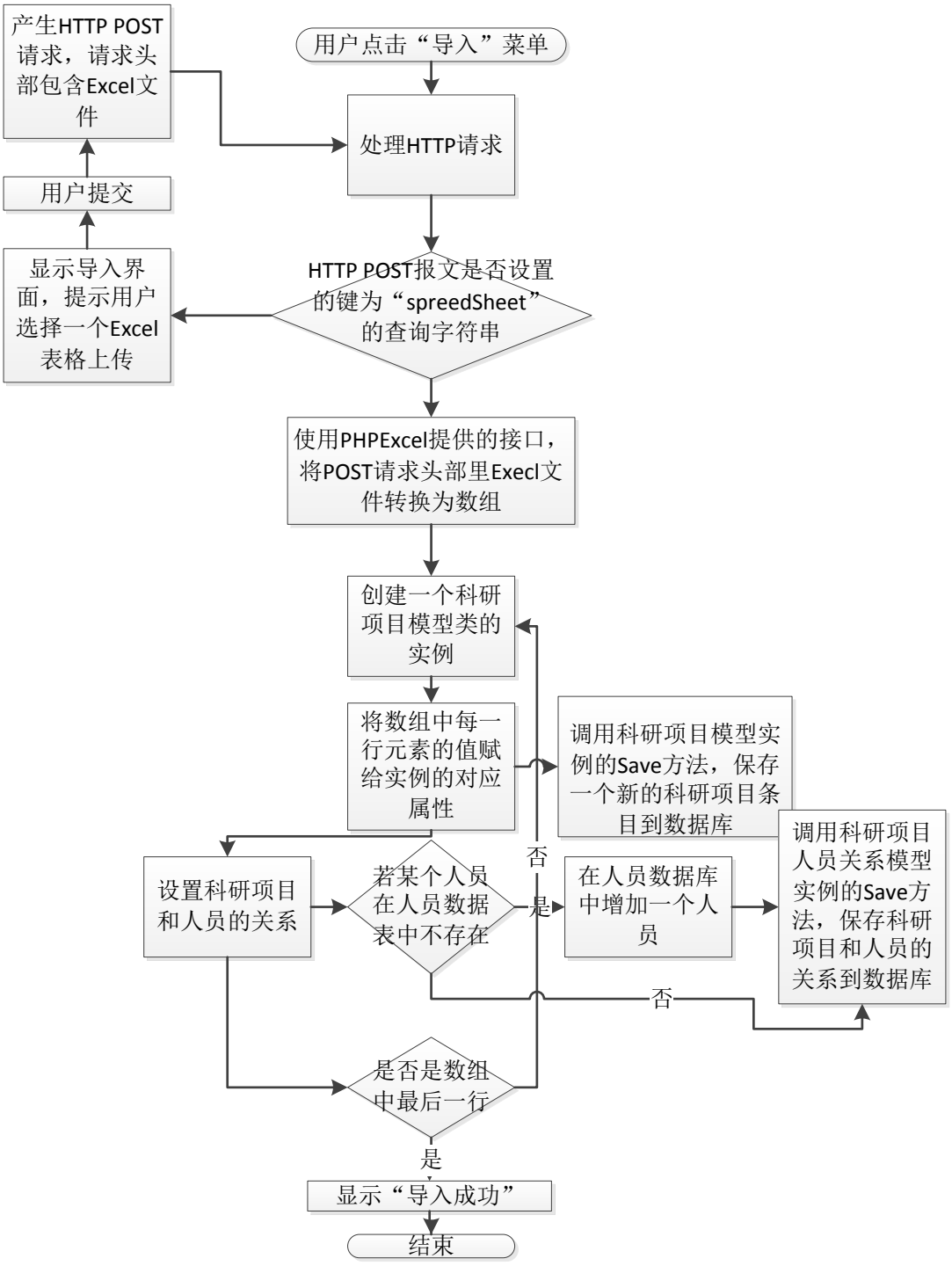


图 5–13 批量录入功能程序流程图

另外，导入数据的Excel表格格式如图 5–14 所示，由于表项太多，在这里仅给出了部分表项，完整的表项见附录 A 。

项目名称	编号	横向	自然	开始时间	人员人1	人员人2	人员人3	人员人4	人员人5	人员人6
面向无线传感器网络的非易失性元胞自动机建模	61104042		是	2012.01.01	于秦	李维乔	王东旭	安宁		
基于分层结构的无线Mesh网络关键技术研究	20080431278			2008.08.01	于秦					
基于元胞自动机的无线传感网络拓扑控制关键技	ZYGX2011J005			2011.07.01	于秦					
异构无线网络的无线资源管理关键技术研究	Y02006023601114			2011.01.01	于秦					
新一代宽带无线网络信道资源管理技术研究	NCET-10-0294			2011.01.01	冷甦鹏					
超高速无线局域网无线接口关键技术研究与验证	2010ZX03005-001			2010.01.01	冷甦鹏	段景山				
无线Mesh网络关键技术研究	L08010101JX05002			2005.10.01	韦云凯	杨建军	吴凡	张科		
基于IPv6的工业WSN组网与节能MAC技术	H04010101W0110119	是		2010.04.01	毛玉明	冷甦鹏	韦云凯	刘强	李龙江	庄奕群
龙门山地震带小流域滑坡泥石流灾害监测预警技	2011BAK12B02			2011.04.01	冷甦鹏	毛玉明	韦云凯	刘强	杨宁	张科
车用无线自组织网络安全告警信息传输策略研究	60802024		是	2009.01.01	冷甦鹏	邵彩幸	汪庆	曾鸣	黄龙娇	刘科
超高速无线局域网无线接口关键技术研究与验证	2010ZX03005-001-01			2010.01.01	冷甦鹏	吴凡	黄晓燕	毛玉明	尹杰晨	邵彩幸
XX抗干扰模型研究	9140C0203010904			2009.05.01	冷甦鹏	于秦	张科	毛玉明	叶景超	王永鑫
车用无线自织网交通安全告警信息传输策略研究	200806141014			2009.01.01	冷甦鹏	邵彩幸	汪庆	曾鸣	黄龙娇	刘科
新世纪优秀人才支持计划项目	NCET-10-0294			2011.01.01	冷甦鹏	吴凡	黄晓燕	张科	杨建军	尹杰晨
先进传感器网络技术的研究	ZX10001	是		2009.12.01	刘强	李龙江	韦云凯	冷甦鹏	毛玉明	庄奕群

图 5–14 导入数据Excel表格格式

5.6.1.2 单个录入

当用户点击“增加科研项目”，Yii框架控制器层的路由功能边将用户点击的URL 解析出来，然后调用相对应的控制器和动作，即控制器 ProjectController 的动作 actionCreate，在这个动作中实现了逐个录入功能。程序的流程图如图 5–15 所示。当首先判断HTTP请求中是否设置在POST报文中设置了键为 Project 的查询字符串，若没有设置，则表示用户未提交新的科研项目条目，返回一个录入界面供用户填写；若设置了，则此次请求是用户在录入界面填写了需要增加的科研项目的对应属性后提交的，分别在数据库中新建一个科研项目的条目到数据库中，并设置对应的关系，以保存科研项目中参与人员的信息。

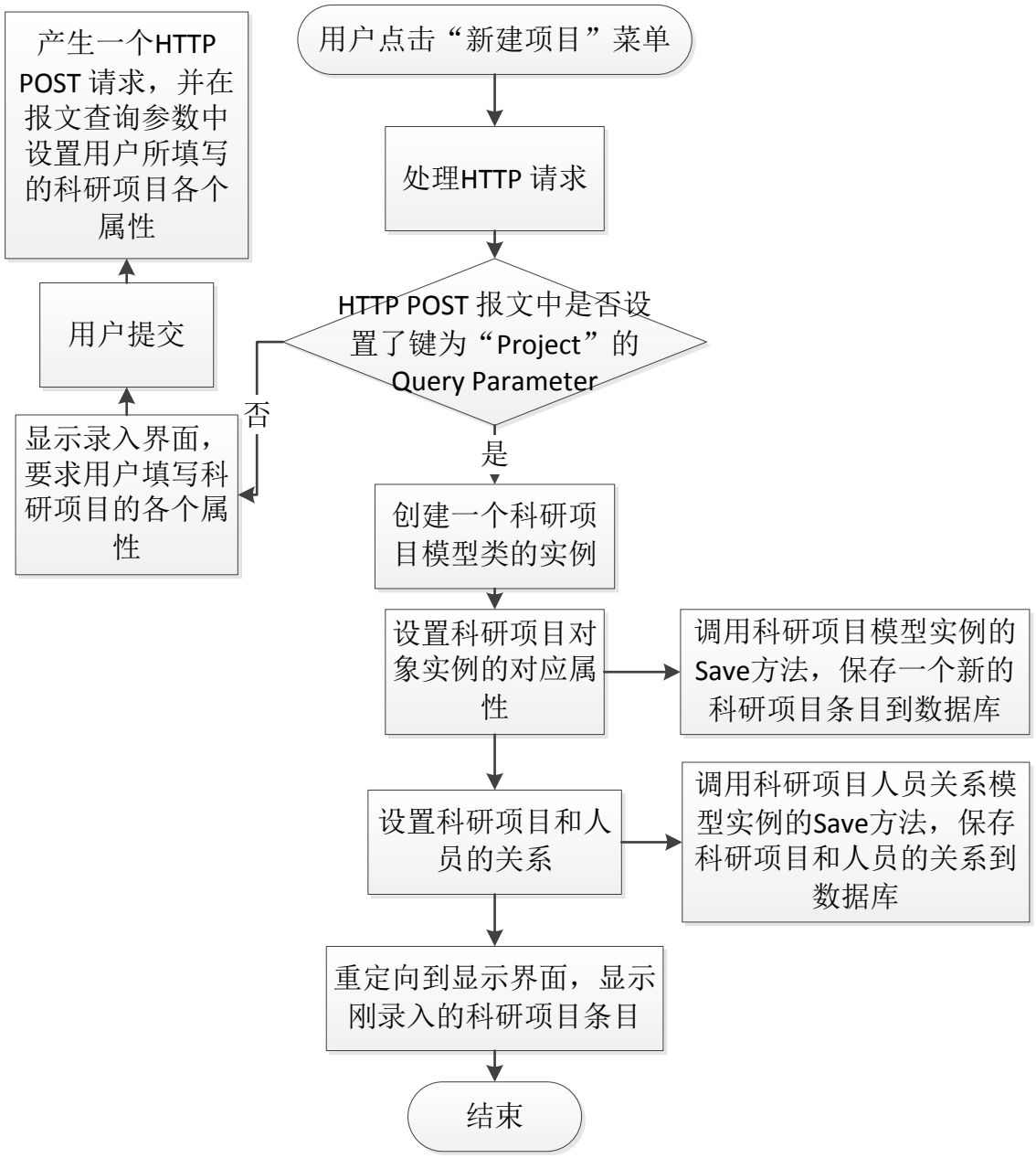


图 5–15 单个录入功能程序流程图

5.6.2 修改

当用户点击“修改科研项目”时，会最终调用到控制器 ProjectController 的动作 actionUpdate，在这个动作中实现了逐个录入功能。修改功能的实现与第 5.6.1 节中逐个录入的实现非常类似，只需在显示修改界面时，读入需要的修改科研项目原来的数据,显示在录入界面的输入控件中。

5.6.3 删除

当用户点击“删除科研项目”时，会最终调用到控制器 ProjectController 的动作 actionDelete，在这个动作中实现了删除某个科研项目的功能功能。程序流程图如图 5–16 所示。

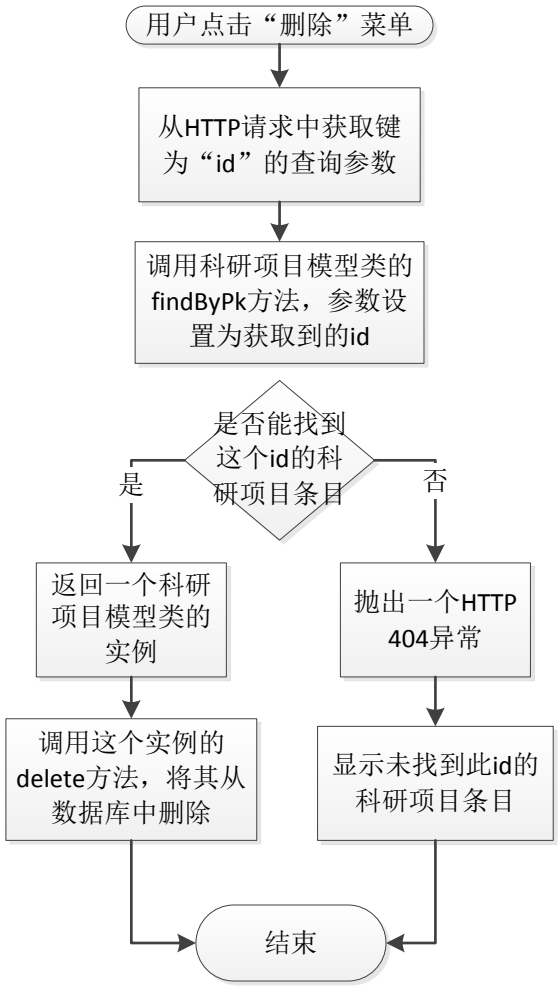


图 5–16 删除功能程序流程图

5.6.4 筛选与查询

当用户点击“管理科研项目”时，会最终调用到控制器 ProjectController 的动作 actionAdmin，在这个动作中实现了逐个筛选与查询。程序流程如下:

1. 例化一个科研项目模型类的实例
2. 按照HTTP请求头部中的查询字符串的键，将HTTP请求头部中的查询字符串的值依次赋给这个实例

3. 调用这个实例的search方法，即在第 46 页第 5.4.6 中实现的方法，返回一个CActiveDataProvider类的实例
4. 将这个CActiveDataProvider类的实例传递给视图层，视图层调用这个实例的 getData 方法，返回所有满足步骤2中筛选条件的研项目模型类的实例，并将它们显示出来。

本系统实现了单个条件以及组合条件进行筛选与查询，例如按日期、按维护人员、按参与者、按时间、按级别等等进行搜索与查询，或者按以上一个条件的组合进行查询。下面以按时间以及级别筛选为例，给出筛选功能的实现，其它条件的实现类似，在这里不再赘述。

1. 如图 5–17 所示，用户希望筛选2010年以后，级别为国家级的项目
2. 通过点击筛选按钮，浏览器发送了如下的查询字符给科研项目控制器（PaperController ）的筛选与查询动作:

   `Project[start_date]:>2010-01-01`

   `Project[is_national]:1`
3. 科研项目控制器的筛选与查询动作(actionSearch)解析这些查询参数，例化一个科研项目模型的实例,将这些条件添加进去
4. 模型类根据这些条件生成SQL语句里WHERE字句的条件：

   ```
   SELECT * FROM ‘tbl_project‘ ‘t‘ WHERE ((is_national=1) AND (↩
   start_date>’2010-01-01’))
   ```
5. 执行上一步生成的SQL，从数据库中取出数据，返回给科研项目控制器的筛选与查询动作
6. 该动作根据管理视图的模版，将这些数据渲染出来，如图 5–18 所示

图 5–17 多条件筛选界面

第 1-3 条, 共 3 条.

项目名称	编号	经费本编号	开始时间	截至时间	结题时间	申报时间	立项时间	申报经费	立项经费	
			>2010-							
基于多信道分配的认知网络信道分配策略研究	61001083		2011-01-01	2013-12-30	2013-12-30			0.00	200000.00	
滑坡泥石流灾害监测预警技术集成与应用示范	2011BAK12B02-6		2010-04-01	2015-03-01	2015-03-01				675000.00	
龙门山地震带小流域滑坡泥石流灾害监测预警技术研究与示范	2011BAK12B02		2011-04-01	2015-03-01	2015-03-01				1260000.00	

图 5–18 多条件筛选结果

第6章 系统测试

本章主要针对设计的系统进行测试，首先给出了测试目标，测试环境等相关内容，接下来，分别从功能上和性能上对本系统进行验证和测试，最后针对测试数据，分析测试结果。

6.1 测试目标

该测试的目标主要分为系统功能和性能测试。性能测试主要测试本系统在低负载和高负载下的吞吐量以及平均响应时间。

6.2 测试环境

在本系统的测试过程中，使用一个服务器和多个客户端进行测试。其中服务器硬件及软件配置如表 6–1 所示，PC客户端硬件及软件配置如表 6–2 所示。除了PC客户端之外，还采用了平板电脑以及手机进行功能测试，测试本系统在不同大小屏幕客户端上的适应能力。

表 6–1 服务器软硬件配置

部件	配置
CPU	Intel Core i5-3470 3.20GHz 4核4线程
内存	8GB
硬盘	2TB 7200转SATA硬盘
操作系统	Ubuntu 12.04 LTS 64-bit
Web服务器	Apache 2.2.22
关系型数据库	MySQL 5.5.32
动态脚本解释器	PHP 5.3.10

表 6–2 PC客户端软硬件配置

部件	配置
CPU	Intel Core i5-430M 2.26GHz 2核4线程
内存	4GB
硬盘	320GB 5400转SATA硬盘
操作系统	Windows 6.1.7601
浏览器	Google Chrome 35.0.1916.114

测试网络环境如图 6–1 所示。

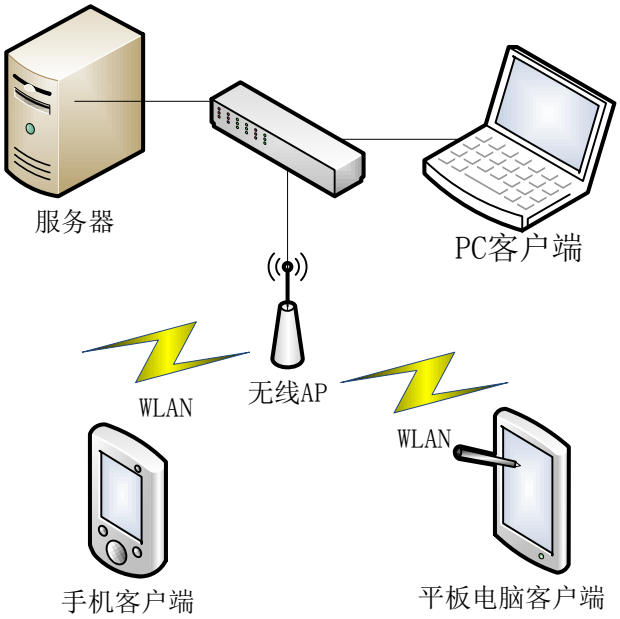


图 6–1 测试网络环境

6.3 功能测试

6.3.1 测试内容

测试内容按找用户身份分为三部分，分别以超级管理员管理员、模块管理员、游客的身份登录系统并测试其各项功能，其中超级管理员管理员、模块管理员可以被称为认证用户，它们都需要测试登录功能。

1. 认证用户

(a) 登录，输入正确的用户名和密码，查看是否能成功登录

(b) 登录，输入错误的用户名和密码，查看是否提示用户名、密码错误

(c) 超级管理员

i. 增加用户，并设置相应权限，查看是否能在数据库增加正确的用户条目

ii. 修改用户的用户名、权限，查看是否能在数据库修改正确的用户条目

iii. 删除用户，查看是否能在数据库删除正确的用户条目

(d) 模块管理员

i. 增加、删除、修改、查询自己所拥有管理权限的模块，查看是否在数据库中有相应的改变

ii. 格式化导入：在自己所拥有管理权限的模块，使用 Excel 文件的数据导入数据，查看数据库中是否正确地导入了Excel文件的数据

iii. 格式化导出：在自己所拥有管理权限的模块，导出满足特定条件的条目为 Excel 文件

iv. 增加、删除、修改、查询自己所不具有管理权限莫模块，应该提示权限不符

2. 游客

(a) 访问各个模块的前台展示部分，查看内容、格式是否正确

(b) 访问系统的后台模块，应该提示权限不符并跳转到登录页面

6.3.2 测试结果

测试的结果最终都是由浏览器的页面显示出来的，所以测试程序是否正确，主要是检查页面是否正确显示。在对本系统进行了全面的测试以后，由于篇幅有限，下面仅举例说明部分关键页面的测试结果。

认证用户登录：输入认证用户正确的用户名和密码，能正确登录并显示出当前用户名和权限，如图 6–2 所示。输入错误的用户名和密码，提示用户名或密码错误错误，如果 6–3 所示。

图 6–2 使用正确的用户名和密码登录

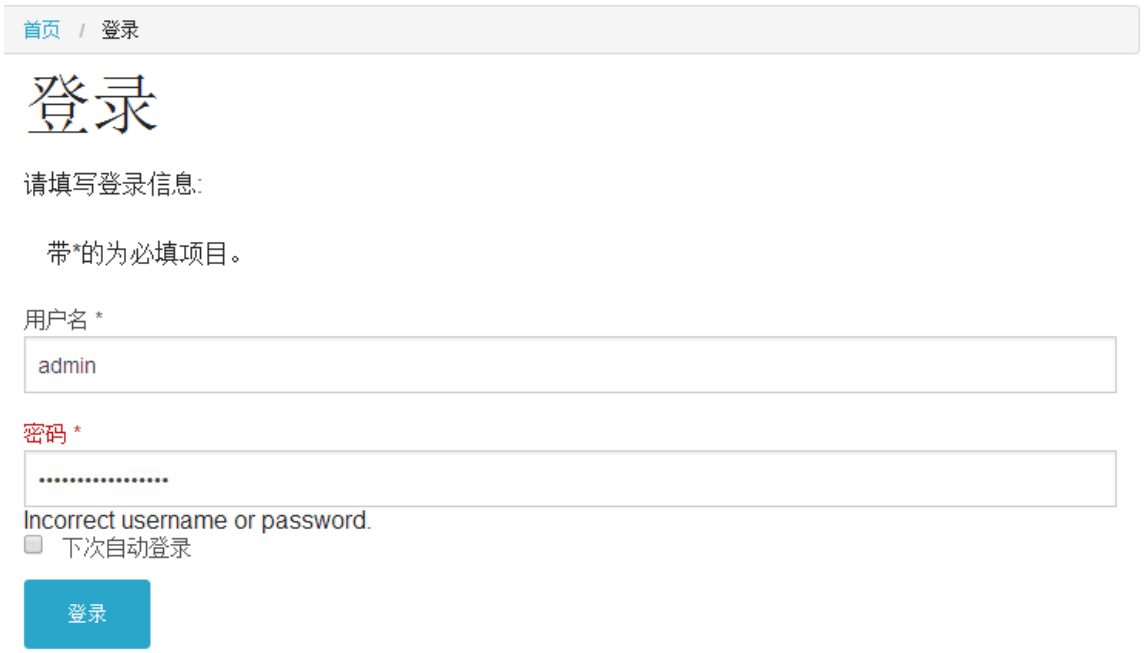


图 6–3 使用错误的用户名和密码登录

增加用户：增加一个新的用户，设置相应权限，如图 6–4 所示，正确的新增了一个用户。

(a) (b)

图 6–4 增加用户

(a)增加用户表单；(b)成功增加用户

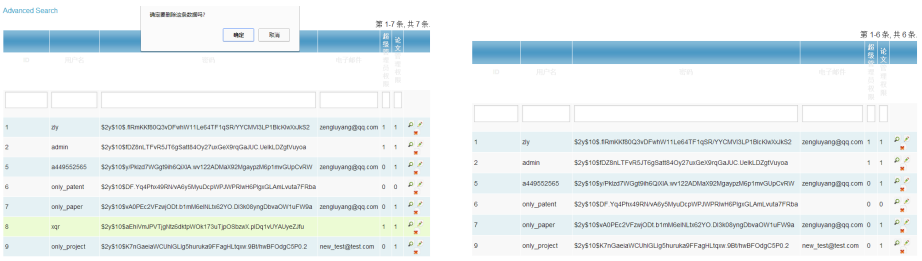
修改用户：修改一个现有的用户，修改后的显示结果与输入的相符合，如图 6–5 所示。

(a) (b)

图 6–5 修改用户

(a)修改用户表单；(b)成功增加用户

删除用户，删除一个现有的用户，删除后该条目从数据库中消失，符合预期结果，如图 6–6 所示。



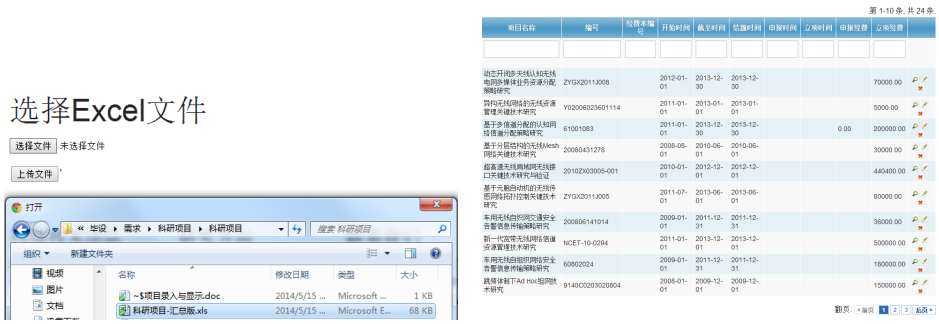
(a) (b)

图 6–6 修改用户

(a)删除用户；(b)成功删除用户

其它数据管理模块增删该查的测试过程类似，且与预期结果相符合，在这里不再赘述。接下来仅介绍科研项目模块的格式化导入、格式化导出功能的测试结果。

科研项目格式化导入：如图 6–7 所示，提交一个在附录 A 给出格式的数据表格，能够正确地将他们提交到数据库中，但在测试过程中发现等待时间较长，大概在10秒左右。后经过分析发现，这是因为使用到的 PHPExcel 库是完全用 PHP 实现的，而PHP是动态解释性语言，并不适合与计算密集型的编程，考虑到使用格式化导入的次数不多，可以等待较长时间，在科研项目控制器类中导入动作中加入“set_time_limit(50);”，将格式化导入功能的执行超时时间设置为50秒，防止导入过程中的超时。



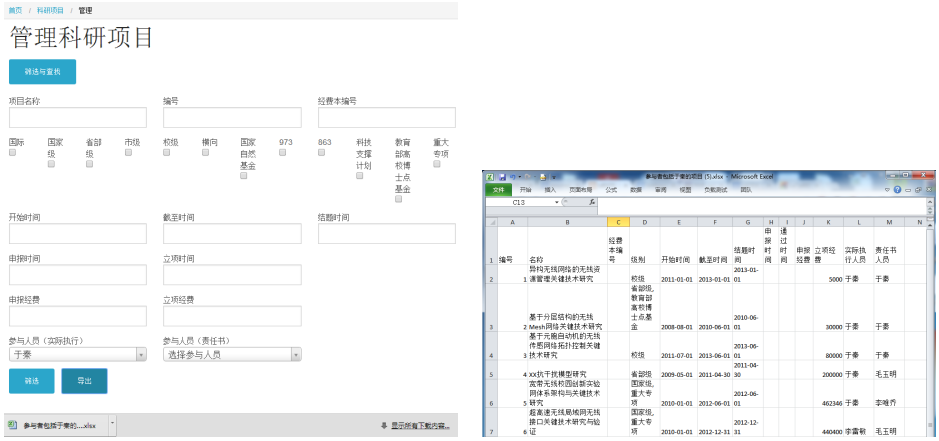
(a) (b)

图 6–7 格式化导如科研项目

(a)选择 Excel 文件；(b)成功导入

科研数据项目格式化导出：如图 6–8 所示，选择筛选条件按人员筛选，能正

确地导出该人员参与的所有科研项目。



(a) (b)

图 6–8 格式化导出科研项目

(a)选择筛选条件；(b)成功出Excel文件

游客访问：如图 6–9所示，能正确地显示相应模块的条目。当游客访问后台模块时，会正确地提示权限不足，如图 6–10所示。

(a) (b)

(c) (d)

图 6–9 游客访问前台展示模块

(a)首页；(b)论文前台显示；(c)科研项目前台显示；(d)专利前台显示

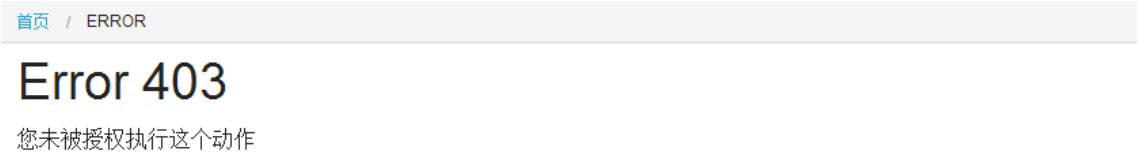
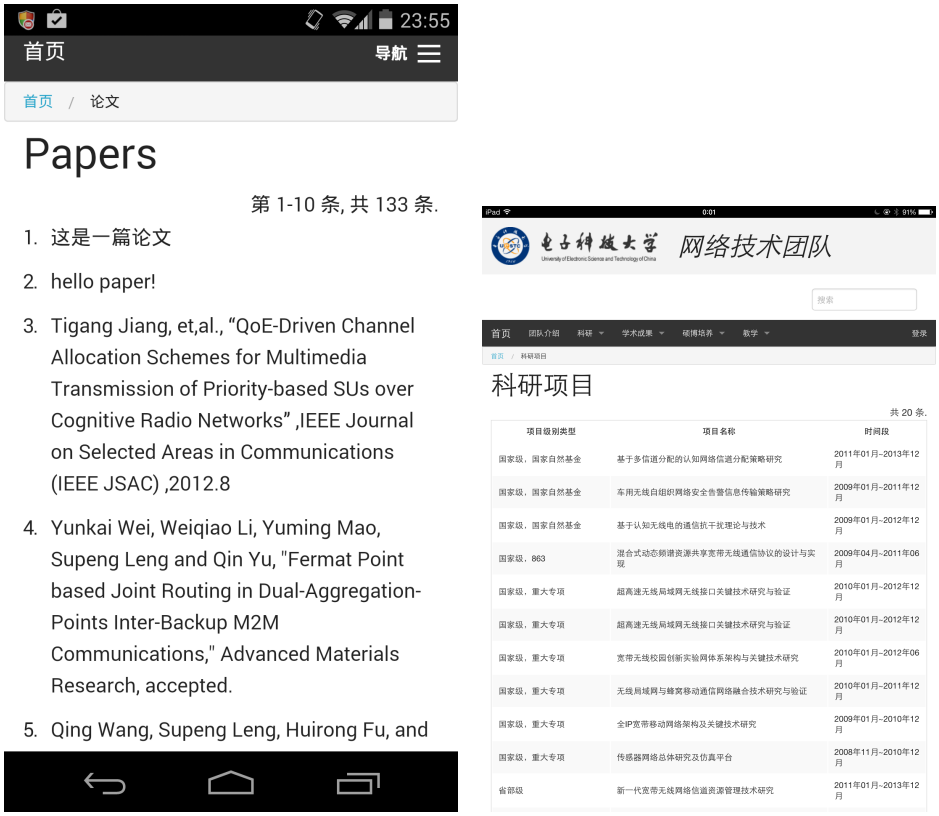


图 6–10 游客访问后台模块，提示权限不足

多客户端响应式布局：分别使用手机和平板电脑访问本系统，可以看到，本系统根据屏幕大小的普通显示内容不同，如图 6–11 所示，在手机上导航栏由下拉菜单改变为了多级菜单方便用户点选，在平板电脑上文字略大，避免用户看不清。



(a) (b)

图 6–11 多客户端响应式布局
(a)使用手机访问本系统；(b)使用平板电脑访问本系统

6.4 性能测试

6.4.1 测试工具

采用 Apache 服务器自带的 Apache Benchmark 工具进行性能测试。Apache Benchmark 工具可以模拟多个用户同时并发访问服务器的情况，对本系统的性能进行黑盒测试。

ApacheBench参数说明:

格式：ab [options] [http://]hostname[:port]/path

-n requests Number of requests to perform：在测试会话中所执行的请求个数（本次测试总共要访问页面的次数）。默认时，仅执行一个请求。

-c concurrency Number of multiple requests to make：一次产生的请求个数（并发数）。

6.4.2 测试内容

本次测试中，使用在表 6–2 中给出的客户端进行测试，服务器和客户端连在同一局域网内，以消除网络的不稳定性对测试的影响，同时能够给予服务器更大的压力。

1. 低负载测试，访问次数为1000，并发数为1，测试命令：
   ab -n 1000 http://192.168.1.10/team/index.php?r=project/index
2. 高负载测试，访问次数为1000，并发数为200，测试命令：
   ab -n 1000 -c 200 http://192.168.1.10 http://192.168.1.10/team/index.php?r=project/index

6.4.3 测试结果

对于低负载测试，测试结果见表 6–3

表 6–3 低负载测试

性能参数	值
吞吐量	19.18（个）
平均响应时间	52.147毫秒

对于高负载测试，测试结果见表 6–4

表 6–4 高负载测试

性能参数	值
吞吐量	55.84（个）
平均响应时间	17.908毫秒

6.4.4 测试结果分析

响应时间是指系统对请求作出响应的时间。直观上看，这个指标与人对软件性能的主观感受是非常一致的，因为它完整地记录了整个计算机系统处理请求的时间。从测试结果中可以看出，本系统的响应时间在10毫秒级别，响应速度较快，具有较高的性能。

吞吐量是指系统在单位时间内处理请求的数量。在并发数200的情形下，本系统的吞吐量为50个左右，能够满足使用需求，考虑到本次测试实在家用PC上完成的，当本系统部署在专用的多核服务器上时，能够达到更高的性能。

第7章 结束语

在课题的方案设计和详细实现中，给出了系统软件的结构和详细设计方法。同时也用在文中提到的开发环境实现了绝大多数的需求，并且经过多次测试，各个功能均能正常运行且没有漏洞，整个系统具有完善的功能、良好的用户体验以及可维护的代码，都达到了预期的期望和课题要求。总的来说本系统具有以下优点：

- 易用性：本系统在各个模块中支持 Excel 数据的批量导入；在下拉菜单中只是拼音首字母的快速筛选；能够各种终端屏幕大小的不同的显示不同的布局。都是考虑到了本系统的操作与使用应该人性化。
- 易维护性：本系统的基础构件简单，易于调整，基于MVC框架进行设计，各模块独立，代码风格良好，易于维护和增加新的功能。
- 性能较好：根据本系统性能测试的结果，本系统部署在家用PC上，每秒钟能够处理50个请求，响应时间在毫秒级，具有较好的并发性与响应速度，若是部署在专用的服务器上，远远达到了本系统日常使用过程中对并发访问以及响应速度的要求。

在本系统的设计与实现中，经历了软件开发过程中完整的瀑布模型[①]，学习和掌握了软件开发的基本流程和方法。同时也通过学习Web开发知识，掌握了Web开发的基本流程和最佳实践。

当然，在未来本系统的实际上线使用过程中，难免会出现一些功能上以及性能的瓶颈，本系统还需要不断地改进与完善。

① 瀑布模型（或称瀑布式开发流程）是由W.W.Royce在1970年首次提出的软体开发模型，在瀑布模型中，软件开发被分为需求分析，设计，实现，测试(确认)，集成，和维护这样的步骤依序进行。

参考文献

[1] J. J. Garrett, et al. Ajax: A new approach to web applications[J]. 2005.

[2] L. Shuchun, L. Mengyang, W. Shixian, et al. The Design and Implementation of the Browser/Server Mode MIS [J][J]. Computer Engineering and Applications, 2000, 6:038.

[3] T. Rentsch. Object oriented programming[J]. ACM Sigplan Notices, 1982, 17(9):51–57.

[4] U. Ramana, T. Prabhakar. Some experiments with the performance of LAMP architecture[C]//Computer and Information Technology, 2005. CIT 2005. The Fifth International Conference on. .[S.l.]: [s.n.] , 2005:916–920.

[5] T. A. S. Foundation. The Apache HTTP Server Project[M].[S.l.]: [s.n.] , 2014. http://httpd.apache.org/.

[6] T. P. Group. PHP 手册 序言[M].[S.l.]: [s.n.] , 2014. http://www.php.net/manual/zh/preface.php.

[7] R. Fielding, J. Gettys, J. Mogul, et al. Rfc 2616, hypertext transfer protocol–http/1.1, 1999[J]. URL http://www. rfc. net/rfc2616. html, 1999.

[8] 蒋磊宏, 董传良, 钟华. 上海交大基于校园网的 MIS 建设规划[J]. 中山大学学报 (自然科学版), 2001, 40(3):10–14.

[9] Z. He-qin. Best scheme of design dynamic website: Apache+ PHP+ MySQL [J][J]. Computer Engineering and Design, 2007, 4:933–938.

[10] T. P. Group. Alternative PHP Cache[M].[S.l.]: [s.n.] , 2014. http://www.php.net/manual/zh/intro.apc.php.

[11] Y. S. LLC. Performance of Yii[M].[S.l.]: [s.n.] , 2010. http://www.yiiframework.com/performance/.

[12] A. Makarov. Yii 1.1 Application Development Cookbook[M].[S.l.]: Packt Publishing Ltd, 2011.

[13] T. Reenskaug. THING-MODEL-VIEW-EDITOR-an Example from a planningsystem[J]. technical note, Xerox Parc, 1979.

[14] 陈旭, 刘加伶. Client/Server 与 Browser/Server 结构的分析与比较[J]. 重庆工学院学报, 2000, 14(2):100–103.

致 谢

历时将近两个月的时间终于将这篇论文写完，在论文的写作过程中遇到了无数的困难和障碍，都在同学和老师的帮助下度过了。尤其要强烈感谢我的论文指导老师—马立香老师，她对我进行了无私的指导和帮助，不厌其烦的帮助进行论文的修改和改进。另外，在校图书馆查找资料的时候，图书馆的老师也给我提供了很多方面的支持与帮助。在此向帮助和指导过我的各位老师表示最衷心的感谢!

感谢这篇论文所涉及到的各位学者。本文引用了数位学者的研究文献，如果没有各位学者的研究成果的帮助和启发，我将很难完成本篇论文的写作。

感谢我的同学和朋友，在我写论文的过程中给予我了很多你问素材，还在论文的撰写和排版等过程中提供热情的帮助。 由于我的学术水平有限，所写论文难免有不足之处，恳请各位老师和学友批评和指正!

附录 A 科研项目批量录入文件格式

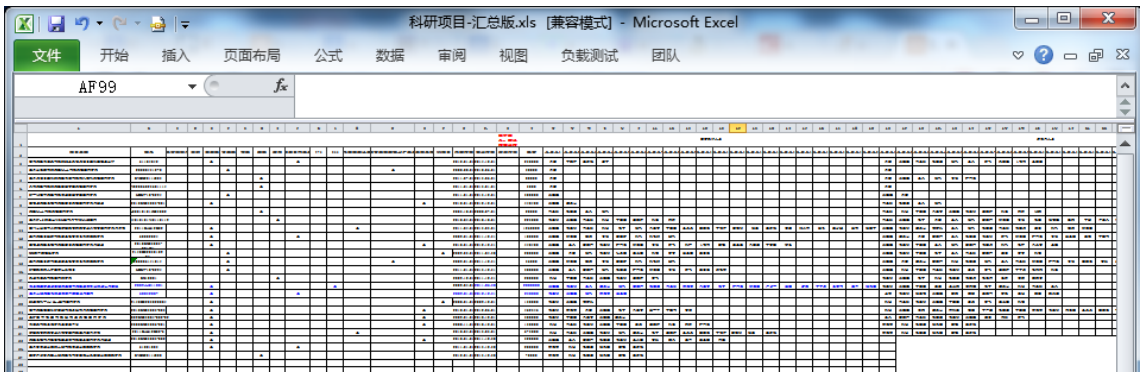


图 A–1 科研项目批量录入Excel文件的格式

如图 A–1 所示，这个录入表格表项很多，有以下表项：

1. 项目名称
2. 经费本编号
3. 是否国际
4. 国家级
5. 省部级
6. 市级
7. 校级
8. 横向
9. 国家自然基金
10. 973
11. 863
12. 科技支撑计划
13. 教育部高校博士点基金
14. 重大专项
15. XX项目
16. 开始时间
17. 截止时间
18. 结题时间

19. 申报时间
20. 立项时间
21. 申报经费
22. 立项经费
23. 实际执行人员1——实际执行人员20
24. 责任书人员1——责任书人员20

附录 B 科研项目管理模块部分源代码

B.1 模型类：Project.php

```
<?php

/**
 * This is the model class for table "tbl_project".
 *
 * The followings are the available columns in table
 'tbl_project':
 * @property integer $id
 * @property string $name
 * @property string $number
 * @property string $fund_number
 * @property integer $is_intl
 * @property integer $is_national
 * @property integer $is_provincial
 * @property integer $is_city
 * @property integer $is_school
 * @property integer $is_enterprise
 * @property integer $is_NSF
 * @property integer $is_973
 * @property integer $is_863
 * @property integer $is_NKTRD
 * @property integer $is_DFME
 * @property integer $is_major
 * @property string $start_date
 * @property string $deadline_date
 * @property string $conclude_date
```

```
27  * @property string $app_date
28  * @property string $pass_date
29  * @property string $app_fund
30  * @property string $pass_fund
31  *
32  * The followings are the available model relations:
33  * @property People[] $tblPeoples
34  */
35 class Project extends CActiveRecord
36 {
37
38
39  const EXECUTE=0;
40     const LIABILITY=1;
41
42
43
44     /**
45      *@var array() of people id
46     */
47  public $executePeoples=array();
48     public $liabilityPeoples=array(); //must be different ↩
49     from the relation name!
50
51
52     public $searchExecutePeople=null;
53     public $searchLiabilityPeople=null;
54
55  /**
56   * @return string the associated database table name
57   */
```

```
58 public function tableName()
59 {
60  return 'tbl_project';
61 }
62
63 public function getLevelList(){
64  return array();
65 }
66
67 public function getTypeString(){
68  return array();
69 }
70
71 /**
72  * @return array validation rules for model attributes.
73  */
74 public function rules()
75 {
76  return array(
77   array('name','required'),
78   array('is_intl, is_national, is_provincial, is_city, ↩
79   is_school, is_enterprise, is_NSF, is_973, is_863, ↩
80   is_NKTRD, is_DFME, is_major', 'numerical', 'integerOnly'=>↩
81   true),
82   array('name, number, fund_number', 'length', 'max'=>255),
83   array('app_fund, pass_fund', 'length', 'max'=>15),
84   array('start_date, deadline_date, conclude_date, app_date,↩
85   pass_date', 'safe'),
86   array('id, name, number, fund_number, is_intl, ↩
87   is_national, is_provincial, is_city, is_school, ↩
88   is_enterprise, is_NSF, is_973, is_863, is_NKTRD, is_DFME, ↩
```

```
89    is_major, start_date, deadline_date, conclude_date, ↩
90    app_date, pass_date, app_fund, pass_fund, ', 'safe', ↩
91    'on'=>'search'),
92   );
93  }
94
95  /**
96   * @return array relational rules.
97   */
98  public function relations()
99  {
100   // NOTE: you may need to adjust the relation name and the ↩
101   related
102   // class name for the relations automatically generated ↩
103   below.
104   /*
105    must add an alias to disambiguate col names in the order ↩
106    clause,
107    can't disambiguate without alias:
108    <WRONG_CODE>
109     ...
110     'execute_peoples' => array(
111               self::MANY_MANY,
112               'People',
113               'tbl_project_people_execute(project_id, ↩
114               people_id)',
115               'order'=>'execute_peoples.seq',
116              ),
117              ...
118             <WRONG_CODE/>
119
```

```
120             seems a bug in the Yii framework, alias seemd to ↩
121             be concated with
122             relationName in the SQL JOIN(instead of ↩
123             replacing it):
124             eg:
125              if not specifying alias, the above WRONG code ↩
126              would render SQL like:
127                LEFT OUTER JOIN `tbl_project_people_execute` ↩
128                `execute_peoples_execute_peoples`
129                ON (`t`.↩
130                `id`=`execute_peoples_execute_peoples`.↩
131                `project_id`)
132              which makes it not possible to disambiguate col ↩
133              names in order clause
134
135   */
136   return array(
137    'liability_peoples' => array(
138     self::MANY_MANY,
139     'People',
140     'tbl_project_people_liability(project_id, people_id)',
141     'order'=>'liability_peoples_liability_.seq',
142     'alias'=>'liability_'
143    ),
144             'execute_peoples' => array(
145              self::MANY_MANY,
146              'People',
147              'tbl_project_people_execute(project_id, ↩
148              people_id)',
149              'order'=>'execute_peoples_execute_.seq',
150              'alias'=>'execute_'
```

```
151             ),
152  );
153
154 }
155
156 /**
157  * @return array customized attribute labels (name=>label)
158  */
159 public function attributeLabels()
160 {
161  return array(
162   'id' => 'ID',
163   'name' => '项目名称',
164   'number' => '编号',
165   'fund_number' => '经费本编号',
166   'is_intl' => '国际',
167   'is_national' => '国家级',
168   'is_provincial' => '省部级',
169   'is_city' => '市级',
170   'is_school' => '校级',
171   'is_enterprise' => '横向',
172   'is_NSF' => '国家自然基金',
173   'is_973' => '973',
174   'is_863' => '863',
175   'is_NKTRD' => '科技支撑计划',
176   'is_DFME' => '教育部高校博士点基金',
177   'is_major' => '重大专项',
178   'start_date' => '开始时间',
179   'deadline_date' => '截至时间',
180   'conclude_date' => '结题时间',
181   'app_date' => '申报时间',
```

```
182     'pass_date' => '立项时间',
183     'app_fund' => '申报经费',
184     'pass_fund' => '立项经费',
185     'execute_peoples' => '实际执行人员',
186     'liability_peoples' => '项目书人员',
187   );
188 }
189
190 /**
191   * Retrieves a list of models based on the current ↩
192   search/filter conditions.
193   *
194   * Typical usecase:
195   * - Initialize the model fields with values from filter ↩
196   form.
197   * - Execute this method to get CActiveDataProvider ↩
198   instance which will filter
199   * models according to data in model fields.
200   * - Pass data provider to CGridView, CListView or any ↩
201   similar widget.
202   *
203   * @return CActiveDataProvider the data provider that can ↩
204   return the models
205   * based on the search/filter conditions.
206   */
207 public function search()
208 {
209
210   $criteria=new CDbCriteria;
211   $criteria->with=array(
212     'execute_peoples',
```

```
213     'liability_peoples'
214   );
215   $criteria->together=true;
216   $criteria->group = 't.id';
217   $criteria->compare('execute_peoples.id',$this->
218   searchExecutePeople,true);
219   $criteria->compare('liability_peoples.id',$this->
220   searchLiabilityPeople,true);
221   $criteria->compare('name',$this->name,true);
222   $criteria->compare('number',$this->number,true);
223   $criteria->compare('fund_number',$this->fund_number,true);
224   $criteria->compare('is_intl',$this->is_intl);
225   $criteria->compare('is_national',$this->is_national);
226   $criteria->compare('is_provincial',$this->is_provincial);
227   $criteria->compare('is_city',$this->is_city);
228   $criteria->compare('is_school',$this->is_school);
229   $criteria->compare('is_enterprise',$this->is_enterprise);
230   $criteria->compare('is_NSF',$this->is_NSF);
231   $criteria->compare('is_973',$this->is_973);
232   $criteria->compare('is_863',$this->is_863);
233   $criteria->compare('is_NKTRD',$this->is_NKTRD);
234   $criteria->compare('is_DFME',$this->is_DFME);
235   $criteria->compare('is_major',$this->is_major);
236   $criteria->compare('start_date',$this->start_date,true);
237   $criteria->compare('deadline_date',$this->deadline_date,
238   true);
239   $criteria->compare('conclude_date',$this->conclude_date,
240   true);
241   $criteria->compare('app_date',$this->app_date,true);
242   $criteria->compare('pass_date',$this->pass_date,true);
243   $criteria->compare('app_fund',$this->app_fund,true);
```

```
244  $criteria->compare('pass_fund',$this->pass_fund,true);
245
246  return new CActiveDataProvider($this, array(
247   'criteria'=>$criteria,
248  ));
249 }
250
251
252 public function getLevelString($glue=', '){
253        $levels = array();
254        $attrs = self::attributeLabels();
255        if($this->is_intl){
256            array_push($levels,$attrs['is_intl']);
257        }
258        if($this->is_national){
259            array_push($levels,$attrs['is_national']);
260        }
261        if($this->is_provincial){
262            array_push($levels,$attrs['is_provincial']);
263        }
264        if($this->is_city){
265            array_push($levels,$attrs['is_city']);
266        }
267        if($this->is_school){
268            array_push($levels,$attrs['is_school']);
269        }
270        if($this->is_enterprise){
271            array_push($levels,$attrs['is_enterprise']);
272        }
273        if($this->is_NSF){
274            array_push($levels,$attrs['is_NSF']);
```

```
275         }
276         if($this->is_973){
277             array_push($levels,$attrs['is_973']);
278         }
279         if($this->is_863){
280             array_push($levels,$attrs['is_863']);
281         }
282         if($this->is_NKTRD){
283             array_push($levels,$attrs['is_NKTRD']);
284         }
285         if($this->is_DFME){
286             array_push($levels,$attrs['is_DFME']);
287         }
288         if($this->is_major){
289             array_push($levels,$attrs['is_major']);
290         }
291
292
293         return implode($glue,$levels);
294
295     }
296
297
298     public function getPeoples($type,$glue=', ',$attr='name')
299     {
300      $peoplesArr = array();
301      switch ($type) {
302       case self::EXECUTE:
303        $peopleRecords=$this->execute_peoples;
304        break;
305       case self::LIABILITY:
```

```
306         $peopleRecords=$this->liability_peoples;
307         break;
308        default:
309         $peopleRecords=$this->execute_peoples;
310         break;
311       }
312       foreach ($peopleRecords as $people) {
313        array_push($peoplesArr,$people->$attr);
314       }
315       return implode($glue,$peoplesArr);
316      }
317
318      public function getExecutePeoples($glue=', ',
319      $attr='name') {
320       return self::getPeoples(self::EXECUTE,$glue,$attr);
321      }
322
323      public function getLiabilityPeoples($glue=', ',
324      $attr='name') {
325       return self::getPeoples(self::LIABILITY,$glue,$attr);
326      }
327
328      public function getExecutePeoplesJsArray($attr='id') {
329       $executePeoples = array();
330       foreach ($this->execute_peoples as $executePeople) {
331        array_push($executePeoples,'"'.$executePeople->$attr.
332        '"');
333       }
334       return implode(', ', $executePeoples);
335      }
336
```

```
337
338     public function getLiabilityPeoplesJsArray($attr='id') {
339      $executePeoples = array();
340      foreach ($this->liability_peoples as $executePeople) {
341       array_push($executePeoples,'"'.$executePeople->$attr.
342       '"');
343      }
344      return implode(', ', $executePeoples);
345     }
346
347 /**
348  * Returns the static model of the specified AR class.
349  * Please note that you should have this exact method in
350  all your CActiveRecord descendants!
351  * @param string $className active record class name.
352  * @return Project the static model class
353  */
354 public static function model($className=__CLASS__)
355 {
356  return parent::model($className);
357 }
358
359
360 protected function beforeSave()
361 {
362  if ($this->pass_date=='')
363            $this->pass_date=null;
364        if ($this->app_date=='')
365            $this->app_date=null;
366        if ($this->conclude_date=='')
367            $this->conclude_date=null;
```

```
368         if($this->scenario=='update') {
369   if(self::deleteProjectPeople(self::EXECUTE) && self::
370   deleteProjectPeople(self::LIABILITY)){
371    return parent::beforeSave();
372   } else {
373    return false;
374   }
375  }
376  return parent::beforeSave();
377 }
378 
379 protected function afterSave() {
380 
381         if(self::populateProjectPeople(self::EXECUTE) && 
382         self::populateProjectPeople(self::LIABILITY)) {
383 
384         }
385         else {
386             return false;
387         }
388 
389     return parent::afterSave();
390    }
391 
392    private function deleteProjectPeople($type) {
393      $criteria = new CDbCriteria;
394        $criteria->condition = 'project_id=:project_id';
395        $criteria->params = array(':project_id'=>$this->id);
396 
397 
398         switch ($type) {
```

```
399             case self::EXECUTE:
400                 ProjectPeopleExecute::model()->deleteAll(
401                 $criteria);
402                 break;
403 
404             case self::LIABILITY:
405                 ProjectPeopleLiability::model()->deleteAll(
406                 $criteria);
407                 break;
408             default:
409                 return ProjectPeopleExecute::model()->
410                 deleteAll($criteria);
411                 break;
412         }
413         return true;
414     }
415     private function populateProjectPeople($type)
416     {
417         switch ($type) {
418             case self::EXECUTE:
419                 $peoples=$this->executePeoples;
420                 break;
421 
422             case self::LIABILITY:
423                 $peoples=$this->liabilityPeoples;
424                 break;
425             default:
426                 $peoples=$this->executePeoples;
427                 break;
428         }
429         for($i=0;$i<count($peoples);$i++) {
```

```
430             if($peoples[$i]!=null && $peoples[$i]!=0) {
431                 switch ($type) {
432                     case self::EXECUTE:
433                         $projectPeople = new ↩
434                         ProjectPeopleExecute;
435                         break;
436                     case self::LIABILITY:
437                         $projectPeople = new ↩
438                         ProjectPeopleLiability;
439                         break;
440                     default:
441                         $projectPeople = new ↩
442                         ProjectPeopleExecute;
443                         break;
444                 }
445 
446                 $projectPeople->seq = $i+1;
447                 $projectPeople->project_id=$this->id;
448                 $projectPeople->people_id=$peoples[$i];
449                 //echo $projectPeople->project_id." ".↩
450                 $projectPeople->people_id."<br>";
451                 if($projectPeople->save()) {
452                     yii::trace("peoples[i]:".$peoples[$i]." ↩
453                     saved","Project.populateProjectPeople()")↩
454                     ;
455                 } else {
456                     return false;
457                 }
458             }
459         }
460         return true;
```

```
461     }
462
463     protected function afterDelete() {
464      if(self::deleteProjectPeople(self::EXECUTE) && self::
465      deleteProjectPeople(self::LIABILITY)){
466    return parent::afterDelete();
467   } else {
468    return false;
469   }
470     }
471 }
```

B.2 控制器类：ProjectController.php

```
1 <?php
2 Yii::import('application.vendor.*');
3 require_once('password_compat/password_compat.php');
4 require_once('PHPExcel/PHPExcel.php');
5 class ProjectController extends Controller
6 {
7  /**
8   * @var string the default layout for the views. Defaults
9   to '//layouts/column2', meaning
10  * using two-column layout. See
11  'protected/views/layouts/column2.php'.
12  */
13  public $layout='//layouts/column2';
14
15  /**
16   * @return array action filters
17   */
```

```
18 public function filters()
19 {
20  return array(
21   'accessControl', // perform access control for CRUD ↩
22   operations
23   //'postOnly + delete', // we only allow deletion via POST ↩
24   request
25  );
26 }
27
28 /**
29  * Specifies the access control rules.
30  * This method is used by the 'accessControl' filter.
31  * @return array access control rules
32  */
33
34 public function accessRules()
35 {
36  return array(
37   array('allow',  // allow all users to perform 'index' and ↩
38   'view' actions
39    'actions'=>array('index','view'),
40    'users'=>array('*'),
41   ),
42   array('allow', // allow authenticated user to perform ↩
43   'create' and 'update' actions
44    'actions'=>array('create','update','admin'),
45    'expression'=>'isset($user->is_project) && $user->↩
46    is_project',
47   ),
48   array('allow', // allow admin user to perform 'admin' and ↩
```

```
49    'delete' actions
50     'actions'=>array('upload','admin','delete','create',↵
51     'update','import','export','pwd','reset'),
52     'expression'=>'isset($user->is_admin) && $user->↵
53     is_admin',
54    ),
55    array('deny',  // deny all users
56     'users'=>array('*'),
57    ),
58   );
59  }
60
61  /**
62   * Displays a particular model.
63   * @param integer $id the ID of the model to be displayed
64   */
65
66  public function actionPwd() {
67   echo "actionPwd.<hr />";
68   $hash1 = password_hash("admin", PASSWORD_DEFAULT);
69   $hash2 = password_hash("123456a", PASSWORD_DEFAULT);
70   var_dump($hash1);
71   var_dump(Yii::app()->user->id);
72   echo $hash1."<hr />";
73   echo $hash2."<hr />";
74   echo password_verify("123456a",$hash1);
75   echo password_verify("123456a",$hash2);
76  }
77
78  public function actionView($id)
79  {
```

```
80   $this->render('view',array(
81    'model'=>$this->loadModel($id),
82   ));
83  }
84
85  private function setModelPeoples($model) {
86   if(isset($_POST['Project']['execute_peoples']))
87    $model->executePeoples=$_POST['Project'][
88    'execute_peoples'];
89   if(isset($_POST['Project']['liability_peoples']))
90    $model->liabilityPeoples=$_POST['Project'][
91    'liability_peoples'];
92  }
93  /**
94   * Creates a new model.
95   * If creation is successful, the browser will be
96   redirected to the 'view' page.
97   */
98  public function actionCreate()
99  {
100   $model=new Project;
101
102   // Uncomment the following line if AJAX validation is
103   needed
104   // $this->performAjaxValidation($model);
105
106   if(isset($_POST['Project']))
107   {
108    $model->attributes=$_POST['Project'];
109    //$model->executePeoples=explode(',',$_POST['Project'][
110    'execute_peoples']);
```

```
111     self::setModelPeoples($model);
112     //var_dump($_POST['Project']['execute_peoples']);
113     //var_dump($model->execute_peoples);
114     //var_dump($_POST['Project']);
115     if($model->save()) {
116      $this->redirect(array('view','id'=>$model->id));
117
118      //var_dump($model->attributes);
119     }
120
121   }
122
123   $this->render('create',array(
124     'model'=>$model,
125   ));
126 }
127
128 /**
129   * Updates a particular model.
130   * If update is successful, the browser will be redirected ⤸
131   to the 'view' page.
132   * @param integer $id the ID of the model to be updated
133   */
134 public function actionUpdate($id)
135 {
136   $model=$this->loadModel($id);
137
138   // Uncomment the following line if AJAX validation is ⤸
139   needed
140   // $this->performAjaxValidation($model);
141
```

```
142   if(isset($_POST['Project']))
143   {
144    $model->attributes=$_POST['Project'];
145    self::setModelPeoples($model);
146    $model->scenario='update';
147    if($model->save())
148     $this->redirect(array('view','id'=>$model->id));
149   }
150
151   $this->render('update',array(
152    'model'=>$model,
153   ));
154  }
155
156  /**
157   * Deletes a particular model.
158   * If deletion is successful, the browser will be ⤶
159   redirected to the 'admin' page.
160   * @param integer $id the ID of the model to be deleted
161   */
162  public function actionDelete($id)
163  {
164   $this->loadModel($id)->delete();
165
166   // if AJAX request (triggered by deletion via admin grid ⤶
167   view), we should not redirect the browser
168   if(!isset($_GET['ajax']))
169    $this->redirect(isset($_POST['returnUrl']) ? $_POST[⤶
170    'returnUrl'] : array('admin'));
171  }
172
```

```
173 /**
174  * Lists all models.
175  */
176 public function actionIndex()
177 {
178  $criteria = new CDbCriteria;
179  $criteria->addCondition('is_school!=1');
180  $dataProvider=new CActiveDataProvider('Project',
181            array('sort'=>array(
182                'defaultOrder'=>array(
183                 'is_intl'=>true,
184                 'is_NSF'=>true,
185                 'is_973'=>true,
186                 'is_863'=>true,
187                 'is_is_NKTRD'=>true,
188                 'is_major'=>true,
189                 'is_provincial'=>true,
190                 'is_city'=>true,
191                 'is_enterprise'=>true,
192                    'start_date' => true,
193                ),
194
195            ),
196            'pagination'=>false,
197            'criteria'=>$criteria
198        ));
199  $this->render('index',array(
200   'dataProvider'=>$dataProvider,
201  ));
202 }
203
```

```
204 /**
205  * Manages all models.
206  */
207 public function actionAdmin()
208 {
209  $model=new Project('search');
210  $model->unsetAttributes();  // clear any default values
211  if(isset($_GET['Project'])) {
212   $model->attributes=$_GET['Project'];
213   if($_GET['Project']['is_intl']=='0') {
214    $model->is_intl="";
215   }
216   if($_GET['Project']['is_national']=='0') {
217    $model->is_national="";
218   }
219   if($_GET['Project']['is_provincial']=='0') {
220    $model->is_provincial="";
221   }
222   if($_GET['Project']['is_school']=='0') {
223    $model->is_school="";
224   }
225   if($_GET['Project']['is_city']=='0') {
226    $model->is_city="";
227   }
228   if($_GET['Project']['is_enterprise']=='0') {
229    $model->is_enterprise="";
230   }
231   if($_GET['Project']['is_NSF']=='0') {
232    $model->is_NSF="";
233   }
234   if($_GET['Project']['is_973']=='0') {
```

```
235     $model->is_973="";
236    }
237    if($_GET['Project']['is_863']=='0') {
238     $model->is_863="";
239    }
240    if($_GET['Project']['is_NKTRD']=='0') {
241     $model->is_NKTRD="";
242    }
243    if($_GET['Project']['is_DFME']=='0') {
244     $model->is_DFME="";
245    }
246    if($_GET['Project']['is_major']=='0') {
247     $model->is_major="";
248    }
249    if(!empty($_GET['People']['execute_id'])){
250     $people=People::model()->findByPk($_GET['People'][↩
251     'execute_id']);
252     $model->searchExecutePeople=$people->id;
253
254    }
255    if(!empty($_GET['People']['liability_id'])){
256     $people=People::model()->findByPk($_GET['People'][↩
257     'liability_id']);
258     $model->searchExecutePeople=$people->id;
259    }
260
261   }
262   //var_dump($model->execute_peoples);
263   $this->render('admin',array(
264    'model'=>$model,
265   ));
```

```
266 }
267
268 /**
269  * Returns the data model based on the primary key given in ⤸
270  the GET variable.
271  * If the data model is not found, an HTTP exception will ⤸
272  be raised.
273  * @param integer $id the ID of the model to be loaded
274  * @return Project the loaded model
275  * @throws CHttpException
276  */
277 public function loadModel($id)
278 {
279  $model=Project::model()->findByPk($id);
280  if($model===null)
281   throw new CHttpException(404,'The requested page does not ⤸
282   exist.');
283  return $model;
284 }
285
286 /**
287  * Performs the AJAX validation.
288  * @param Project $model the model to be validated
289  */
290 protected function performAjaxValidation($model)
291 {
292  if(isset($_POST['ajax']) && $_POST['ajax']==='project-⤸
293  form')
294  {
295   echo CActiveForm::validate($model);
296   Yii::app()->end();
```

```
297   }
298 }
299
300     public function actionUpload() {
301         set_time_limit(50);
302         if(isset($_FILES['spreedSheet']) && !empty($_FILES[↩
303         'spreedSheet'])) {
304             $path = $_FILES['spreedSheet']['tmp_name'];
305             echo $_FILES['spreedSheet']['name']."<hr />";
306             echo $_FILES['spreedSheet']['type']."<hr />";
307             echo $_FILES['spreedSheet']['tmp_name']."<hr />";
308             if(self::saveXlsToDb($path)){
309                 echo 'function actionUpload() succeeded.<hr ↩
310                 />';
311                 $this->redirect(array('index'));
312             }
313         }
314
315
316         $this->render('upload');
317     }
318
319     public function actionReset() {
320         Project::model()->deleteAll();
321     }
322
323
324
325
326     protected function saveXlsToDb($xlsPath) {
327         $projects = self::xlsToArray($xlsPath);
```

```
328         return self::saveXlsArrayToDb($projects);
329     }
330 
331     public function xlsToArray($path)
332     {
333         Yii::trace("start of loading","actionTestXls()");
334         $reader = PHPExcel_IOFactory::createReader('Excel5');
335         $reader->setReadDataOnly(true);
336         $objPHPExcel = $reader->load($path);
337         Yii::trace("end of loading","actionTestXls()");
338         Yii::trace("start of reading","actionTestXls()");
339         $dataArray = $objPHPExcel->getActiveSheet()->toArray(
340         null,true,true);
341         Yii::trace("end of reading","actionTestXls()");
342         array_shift($dataArray);
343         //var_dump($dataArray);
344         return $dataArray;
345     }
346 
347     private function convertYesNoToInt($yesno) {
348         if($yesno=='是') {
349             return 1;
350         }else if($yesno=='否'){
351             return 0;
352         }
353         return 0;
354     }
355 
356 
357 
358     public function saveXlsArrayToDb($projects)
```

```
359     {
360         $connection=Yii::app()->db;
361         //var_dump($projects);
362         foreach($projects as $k => $p) {
363             //var_dump($k);
364             //var_dump($p);
365             if($k<2) continue;
366             $project = new Project;
367             $project->name=$p[0];
368             $project->number=$p[1];
369             $project->fund_number=$p[2];
370             $project->is_intl=self::convertYesNoToInt($p[3]);
371             $project->is_national=self::convertYesNoToInt($p[
372             4]);
373             $project->is_provincial=self::convertYesNoToInt(
374             $p[5]);
375             $project->is_city=self::convertYesNoToInt($p[6]);
376             $project->is_school=self::convertYesNoToInt($p[7]
377             );
378             $project->is_enterprise=self::convertYesNoToInt(
379             $p[8]);
380             $project->is_NSF=self::convertYesNoToInt($p[9]);
381             $project->is_973=self::convertYesNoToInt($p[10]);
382             $project->is_863=self::convertYesNoToInt($p[11]);
383             $project->is_NKTRD=self::convertYesNoToInt($p[12]
384             );
385             $project->is_DFME=self::convertYesNoToInt($p[13])
386             ;
387             $project->is_major=self::convertYesNoToInt($p[14]
388             );
389             $project->start_date=($p[16]);
```

```
390             $project->deadline_date=($p[17]);
391             $project->conclude_date=empty($p[18]) ? $project->
392             >deadline_date : $p[18];
393             $project->pass_fund=$p[19];
394             //if($project->save()) {
395                 $peoplesId=array();
396                 for($i=0;$i<20;$i=$i+1){
397                     $peopleName=$p[20+$i];
398                     $peopleName=mysql_real_escape_string(
399                     $peopleName);
400                     $sql='select id from tbl_people where 
401                     name="'.$peopleName.'";';
402 
403                     //$testPeoples = People::model()->find(
404                     'name="'.$peopleName.'"');
405                     //var_dump($testPeoples);
406                     $command=$connection->createCommand($sql)
407                     ;
408                     $row=$command->queryRow();
409                     if($row) {
410                         //dump($row);
411                         $peoplesId[]=$row['id'];
412 
413                     }else {
414                         //dump($row);
415                         $people = new People;
416                         $people->name = $peopleName;
417                         if($people->save())
418                             //dump($people->id);
419                         $peoplesId[] = $people->id;
420                     }
```

```
421
422                 }
423                 //if(!self::populatePeople($project,
424                 $peoplesId))
425                 //    return false;
426                 $project->executePeoples = $peoplesId;
427                 $peoplesId=array();
428                 for($i=0;$i<20;$i=$i+1){
429                     $peopleName=$p[40+$i];
430                     $peopleName=mysql_real_escape_string(
431                     $peopleName);
432                     $sql='select id from tbl_people where
433                     name="'.$peopleName.'";';
434
435                     //$testPeoples = People::model()->find(
436                     'name="'.$peopleName.'"');
437                     //var_dump($testPeoples);
438                     $command=$connection->createCommand($sql)
439                     ;
440                     $row=$command->queryRow();
441                     if($row) {
442                         //dump($row);
443                         $peoplesId[]=$row['id'];
444
445                     }else {
446                         //dump($row);
447                         $people = new People;
448                         $people->name = $peopleName;
449                         if($people->save())
450                             //dump($people->id);
451                         $peoplesId[] = $people->id;
```

```
452                     }
453
454                 }
455                 $project->liabilityPeoples = $peoplesId;
456                 $project->save();
457
458             //} else {
459             //    var_dump( $project->getErrors());
460                 //var_dump($project->info);
461             //    return false;
462             //}
463         }
464         return true;
465     }
466
467
468
469 }
```

Best MVC Practices

Although Model-View-Controller (MVC) is known by nearly every Web developer, how to properly use MVC in real application development still eludes many people. The central idea behind MVC is code reusability and separation of concerns. In this section, we describe some general guidelines on how to better follow MVC when developing a Yii application.

To better explain these guidelines, we assume a Web application consists of several sub-applications, such as

- front end: a public-facing website for normal end users; back end: a website that exposes administrative functionality for managing the application. This is usually restricted to administrative staff;
- Tconsole: an application consisting of console commands to be run in a terminal window or as scheduled jobs to support the whole application; Web API: providing interfaces to third parties for integrating with the application.

The sub-applications may be implemented in terms of modules, or as a Yii application that shares some code with other sub-applications.

1.1 Model

Models represent the underlying data structure of a Web application. Models are often shared among different sub-applications of a Web application. For example, a LoginForm model may be used by both the front end and the back end of an application; a News model may be used by the console commands, Web APIs, and the front/back end of an application. Therefore, models

Models represent the underlying data structure of a Web application. Models are often shared among different sub-applications of a Web application. For example, a LoginForm model may be used by both the front end and the back end of an application; a News model may be used by the console commands, Web APIs, and the front/back end of an application. Therefore, models

- should contain properties to represent specific data;
- should contain business logic (e.g. validation rules) to ensure the represented data fulfills the design requirement;
- may contain code for manipulating data. For example, a SearchForm model, besides representing the search input data, may contain a search method to implement the actual search.

Sometimes, following the last rule above may make a model very fat, containing too much code in a single class. It may also make the model hard to maintain if the code it contains serves different purposes. For example, a News model may contain a method named getLatestNews which is only used by the front end; it may also contain a method named getDeletedNews which is only used by the back end. This may be fine for an application of small to medium size. For large applications, the following strategy may be used to make models more maintainable:

- Define a NewsBase model class which only contains code shared by different sub-applications (e.g. front end, back end);
- In each sub-application, define a News model by extending from NewsBase. Place all of the code that is specific to the sub-application in this News model.

So, if we were to employ this strategy in our above example, we would add a News model in the front end application that contains only the getLatestNews method, and we would add another News model in the back end application, which contains only the getDeletedNews method.

In general, models should not contain logic that deals directly with end users. More specifically, models

- should not use $ _GET, $ _POST, or other similar variables that are directly tied to the end-user request. Remember that a model may be used by a totally different sub-application (e.g. unit test, Web API) that may not use these variables to represent user requests. These variables pertaining to the user request should be handled by the Controller.
- should avoid embedding HTML or other presentational code. Because presentational code varies according to end user requirements (e.g. front end and back end may show the detail of a news in completely different formats), it is better taken care of by views.

1.2 View

Views are responsible for presenting models in the format that end users desire. In general, views

- should mainly contain presentational code, such as HTML, and simple PHP code to traverse, format and render data;
- should avoid containing code that performs explicit DB queries. Such code is better placed in models.
- should avoid direct access to $ _GET, $ _POST, or other similar variables that represent the end user request. This is the controller's job. The view should be focused on the display and layout of the data provided to it by the controller and/or model, but not attempting to access request variables or the database directly.
- may access properties and methods of controllers and models directly. However, this should be done only for the purpose of presentation.

Views can be reused in different ways:

- Layout: common presentational areas (e.g. page header, footer) can be put in a layout view.
- Partial views: use partial views (views that are not decorated by layouts) to reuse fragments of presentational code. For example, we use _form.php partial view to render the model input form that is used in both model creation and updating pages.
- Widgets: if a lot of logic is needed to present a partial view, the partial view can be turned into a widget whose class file is the best place to contain this logic. For widgets that generate a lot of HTML markup, it is best to use view files specific to the widget to contain the markup.
- Helper classes: in views we often need some code snippets to do tiny tasks such as formatting data or generating HTML tags. Rather than placing this code directly into the view files, a better approach is to place all of these code snippets in a view helper class. Then, just use the helper class in your view files. Yii provides an example of this approach. Yii has a powerful CHtml helper class that can produce commonly used HTML code. Helper classes may be put in an autoloadable directory so that they can be used without explicit class inclusion.

1.3 Controller

Controllers are the glue that binds models, views and other components together into a runnable application. Controllers are responsible for dealing directly with end user requests. Therefore, controllers

- may access $ _GET, $ _POST and other PHP variables that represent user requests;
- may create model instances and manage their life cycles. For example, in a typical model update action, the controller may first create the model instance; then populate the model with the user input from $ _POST; after saving the model successfully, the controller may redirect the user browser to the model detail page. Note that the actual implementation of saving a model should be located in the model instead of the controller.
- should avoid containing embedded SQL statements, which are better kept in models.
- should avoid containing any HTML or any other presentational markup. This is better kept in views.

In a well-designed MVC application, controllers are often very thin, containing probably only a few dozen lines of code; while models are very fat, containing most of the code responsible for representing and manipulating the data. This is because the data structure and business logic represented by models is typically very specific to the particular application, and needs to be heavily customized to meet the specific application requirements; while controller logic often follows a similar pattern across applications and therefore may well be simplified by the underlying framework or the base classes.

MVC的最佳实践

虽然几乎每个 Web 开发者都知道模型-视图-控制器架构， 但是如何在实际的应用开发中使用 MVC 仍然难倒了很多人。蕴含在MVC中的主要思想是代码的可重用性和关注点分离[①]。在本章中，将描述一些在开发 Yii 应用时如何更好地运用MVC的指引。

为了更好的阐述这些指引，我们假定一个Web应用包括了一个子应用，比如：

- 前台：一个为普通最终用户所使用公开的网站；
- 后台：一个提供管理功能的网站。通常只能由管理用户访问；
- 控制台：一个又控制台命令构成的应用，在终端窗口中所使用或者以计划任务的方式使用，以一尺整个应用；
- Web API：为第三方程序的整合提供接口。

这些子应用可以以模块的方式实现，或者以与其它子应用Yii应用的方式实现。

1.1 模型

模型代表支撑Web应用的数据结构。模型通常可以被Web应用不同的子应用间所共用。例如，一个登录表单的模型可以被一个应用前台的前台和后台所使用；一个新闻模型可以被一个应用的控制台、Web API以及前后台所使用。因此，模型应该

- 包括表示特定数据的属性；
- 包括特定业务逻辑（比如说，验证规则）来确保它所表示的数据满足设计要求；
- 可以包括操作数据的代码。比如说，一个搜索表单模型，除了表示搜索输入数据外，可能包含一个函数实现搜索的功能。

① 译者注：关注点分离（Separation of concerns，SOC）是对只与“特定概念、目标”(关注点)相关联的软件组成部分进行“标识、封装和操纵”的能力，即标识、封装和操纵关注点的能力。是处理复杂性的一个原则。由于关注点混杂在一起会导致复杂性大大增加，所以能够把不同的关注点分离开来，分别处理就是处理复杂性的一个原则，一种方法。

有时候，遵循上述最后一条规则可能会导致一个模型非常臃肿，在一个类中包含过多的代码。也可能导致这个模型很难维护，如果它完成的很多作用。例如，一个新闻模型可能包含了一个名为getLatestNews（获取最新新闻）的函数，这个函数只在前台使用；它也可能包含一个名为getDeletedNews（获取已删除新闻）的函数，这个函数只被后台所使用。对于一个中小型的网站来说，这还问题不大。但对于大型网站，下列策略会让模型更加易维护：

- 定义一个新闻基类，这个类只包含各个子应用（例如：前台，后台）所共用的代码；
- 在每个子应用中，通过继承新闻基类，来定义一个新的新闻类，将所有对与各个子应用特定的代码放在里面。

这样一来，如果把这个策略使用到上面的例子当中，便应该在前台中添加只包括getLatestNews（获取最新新闻）函数的新闻模型类，在后台中添加另外一个只包括getDeletedNews（获取已删除新闻）函数的新闻模型类。

总的来说，模型不应该包括直接与最终用户有关的逻辑。具体地说，模型：

- 不应该使用$ _GET, $ _POST，或者其它与最终用户请求直接有关的变量。记住模型可能被完全不同的子程序（例如：单元测试，Web API）所使用，这些子程序可能不用这些变量来表示用户请求。这些与最终用户请求有关的变量应该被控制器处理。
- 应该避免嵌入HTML或者其他表示层代码。因为表示层的代码会根据最终用户不同的要求所变化（例如，前台和后台可能以完全不同的形式来展示一条新闻的细节），这些表示层代码最好由视图来处理。

1.2 视图

视图负责按照最终用户要求的格式将模型展示出来。中的来说，视图：

- 应该主要包含表示层代码，比如说HTML，以及简单的PHP代码来遍历、格式化以及渲染数据；
- 应该避免包含直接执行数据库查询的代码，这些代码放在模型里更好。
- 应该不免直接访问 $ _GET, $ _POST，或者其他表示最终用户请求的类似变量。这是控制器的工作。视图应该主要处理控制器和模型提供的数据，将它们按照特定的格式展示出来，而不是尝试直接访问用户请求变量或者数据库。

- 可以直接访问模型和控制器的属性与函数。但是，只能在完成表示目的时，执行这些操作。

视图可以按照一下的方式被重用：

- 共同的表示区域（例如，页面的头部、尾部）可以被放在一个布局视图里。
- 部分视图：使用部分视图（不被布局所装饰的视图）来重用表示层的代码段。不如说，使用 _form.php 部分视图来渲染增加和修改页面都用到的输入表单。
- 控件：如果在渲染一个部分视图时需要很多的逻辑，这个部分视图可以被封装成一个空间，这个控件的类文件是防止这些逻辑代码最好的地方。对于那些需要产生很多HTML标记的空间，最好能够使用对于这些空间来说特定的视图文件来防止这些标记。
- 助手类：在试图中会经常使用到一些代码端来完成一些小的任务，比如说格式化数据或者产生HTML标签。与其直接将这些代码放在视图文件里，一个跟好的方法是将这些代码放在视图助手类里面。 Yii 包含了使用这种方法的例子。Yii有一个强大的CHtml助手类，这个类可以产生常用的HTML代码。助手类可以被放在自动载入的目录里，这样的话他们无需显示的类包含声明便可以被直接使用。

1.3 控制器

控制器是将模型、视图以及其他组将绑定为一个可运行应用的胶水。控制器负责直接处理最终用户请求。因此，控制器

- 可以访问 $ _GET 、 $ _POST 以及其它表示用户请求的PHP变量；
- 可以创建模型类的实例和管理它们的生命周期。例如，在一个典型的模型修改动作中，控制器可以首先创建模型实例；然后将从 $ _POST 中得到的用户输入填入到这个模型实例中；在成功保存了这个模型实例之后，控制器可以将用户的浏览器重定向至这个模型的详细介绍页面。需要注意的是，保存一个模型的实际实现应该放在模型里面，而不是控制器里。
- 应该避免包含直接执行数据库查询的代码，这些代码放在模型里更好;
- 应该避免嵌入HTML或者其他表示层代码，这些表示层代码最好由视图来处理。

在一个设计完善的MVC应用中，控制器通常都很稀薄，仅仅包含数百行代码；模型通常比较臃肿，包含了大部分表示和操作数据的代码。这是因为数据结构和业务逻辑通常是和特定应用的需求紧密联系在一起的；与此不同，控制器的逻辑在不同的应用间通常有着一种类似的模式，因此能够被底层的框架或者基类所简化。