基于python的大学生云课堂平台设计

# 摘要**：**随着信息技术的发展，互联网提供给用户的服务越来越多。就教学方式来说，国内传统的线下的教学模式已经变得不再单一，网上出现了很多供学生自主学习的云课堂平台。

# 本文将围绕这一现象进行设计并探讨如何实现一个基于B/S数据库架构的分布式云课堂开发平台,利用目前比较流行的Python语言上的Django数据库框架与MySQL数据库框架进行设计和开发。此平台对于云课堂学生、老师、管理员三个主要管理模块的开发和设计,旨在能够让教师、学生在分布式云课堂下能够有着更进一步的学习和交流。 本文中的平台主要完成了登陆注册模块、课程模块、讲师模块、个人中心模块和后台管理模块的设计与实现。同时本系统界面整洁、操作方便，通过接入X-admin后台管理框架使得系统具有良好的维护性和可扩展性。

**关键词：**Python, Django，云课堂，MySQL

# **The programming of online-class platoon for college students based on python**

**Abstract:**With the development of information technology, the internet provides more and more services to its users. In terms of the teaching methods, the traditional offline teaching mode is not the only way, plenty of new online class platforms that offer students to learn by themselves are becoming more and more popular.

With the inspiration of this phenomenon, the author discusses in this article about how to design a distributed cloud classroom platform based on B/S database architecture. The entire design and development of the platform were by using the Django database frame which from the most popular programming language Python and MySQL database frame. To achieve the goal of providing teachers and students a better way to study and communicate in this distributed cloud classroom, the design and development of this platform are oriented to three different groups of people: students, teachers, and administrators. Main modules including registration and login module, course module, lecturer module, personal center module, and backstage management module are designed are and implemented. Meanwhile, the interface of this system is concise and user-friendly. Maintainability and expandability of the system are highly improved by connecting with X-admin backstage management frame.

**Key words:**Python, Django, online-class, MySQL

# 第1章 绪 论

## 1.1 课题的背景与意义

现在，随着网络的飞速发展，老百姓使用网络的频率也越来越高[1]。截止至2019年中旬,我国计算机网民的规模已经高达八亿余人。同时，我国颁布了“云计算发展计划”，进一步促进以“人”为本的云课堂发展，在高校普及率呈高速增长趋势[1]。目前我国移动计算机的信息化普及和发展速度非常之快,计算机以及移动网络也在飞速发展,同时,互联网的普及和发展也直接带动了我国网络教育和信息化的进一步发展。目前，大部分的学校已经实现了网络全校覆盖，基本上所有师生都能实现上网需求，再加上现在智能手机的普及，云课堂的实现具备了更强的现实基础。

## 1.2 课题发展现状和前景展望

随着互联网信息化的深入渗透到我们的工作和日常生活,越来越多的年轻人开始意识到它给我们的日常生活工作和学习生活带来了巨大的便利和无限的发展潜力。而对于很多高校的毕业生来讲,学校在信息网络的建设和管理方面已经在国际上有了突飞猛进的快速发展,我们高校就应该合理的利用这一点,而合理利用云课堂学习平台的优点是能够方便我们教师的教学和有利于学生的学习,有利于提高教师的教学工作效率和对学生的学习活动效率的培养和提高。云课堂的平台里面以视频教学为主,同时也很充分的注重课堂师生间的互动和沟通交流,可以有效拉近老师与其他学生之间的关系和距离,让更多的学生能够自愿、轻松的参与课堂学习。它的主要几个优点同样具有:

1.老师能够通过线上实时了解和跟进学生的学习情况，做到对学生进行学习督促、学习教导等。

2.学生不再只局限于以前在教室上课时学到的知识，学生能共享老师在课程列表发布的课程资，还能在线上与老师同学进行讨论交流。同时学生还能通过讲师在课前发布的注意事项对课程提前进行预习，能使学生在线下直播听课学习效率更高。

3.老师能随时对学生进行线上测验和抽查，来督促学生在家里也能高效学习。同时线上的限时测试能避免学生之间的抄袭现象，能更好的引导学生独立自主的完成课堂测验和作业。

4.方便老师对学生的日常学习和测验数据进行统计分析，从而实现“因材施教”。

毋庸置疑云课堂可以加强学生和学生之间的关系，能改变单一的线下直播授课教学高节奏、无监管、学生效率低下的问题。以课堂面对面为主、以网络课堂为辅的教学方式更有利于大学生的学习[1]。

## 1.3 课题主要内容和要求

本次课题中设计的系统主要实现的要求有如下:

(1)用户能完成基本的登陆注册功能。

(2)学生能在个人中心里完成学生信息管理，有添加和随时修改学生信息的功能，在我的课堂里能查看已学习课程列表以及其他课程信息。

(3)用户能查找到自己想听的老师的课。

(4)平台能提供找课、看课、学课的一站式服务，用户能查看课程列表，查看课程简介以及详情，下载相关素材和视频进行预习和复习。

(5)管理员能在后台管理完成对任意课程的增加、修改和删除的功能。

## 1.4 论文的工作和安排

本次论文主要用到了python语言，并用web框架Django和MySQL数据库进行开发。设计的目标是建立一个大学生云课程平台的网站，同时网站要实现基本的登陆注册、个人信息的修改和找课看课的功能。具体的研究步骤如下：

第一章是绪论，主要介绍了云课堂平台课题的相关背景和发展前景。

第二章是本次开发所要用的开发工具的介绍，本章主要介绍的内容是基础软件开发应用工具及与基础软件开发相关的一些关键技术和几种开发环境。

第三章是对云课堂系统的整体功能结构的系统需求分析，这一章主要内容是详细介绍了云课堂系统的各个主要组成功能模块的整体需求结构分析以及概述。

第四章主题是云课堂系统的设计,本章主要是介绍了系统的架构以及整体设计框架和对于用户体验模块、云课堂模块、讲师的管理模块以及云课堂后台的管理模块给予了详细的介绍和设计以及系统源代码的阐述。

第五章主要内容是关于代码管理系统的功能实现,本书各章节所有主要功能模块的代码系统结构功能实现视图都对此进行了详细的分析阐述,并分别重点给出了具体的各种代码系统结构设计和代码系统功能实现的整体效果图。

# 第2章 开发工具及相关技术

## 2.1 Python

Python是一种具有很多功能的计算机程序设计语言[2]。1991年Python第一次发布。在2004年以后,Python的源代码使用率已经呈现出了迅速的增长。2008年12月3日发布了Python3.0版本,这代表着Python语言的重大跨越,Python3的发行重新修正了之前其他Python版本的一些程序语法的缺陷。Python3开发的主要重点之一就是清理并删除重复、多余的代码数据，并明确使用一种方法来实现程序。Python3无法进行向下兼容,也就是说所有Python2的程序代码在所有python3中不可能支持向下运行。从对所有Python3提供向下支持的所有Python包的程序代码数量中我们可以很清楚的看出,Python3已经开始得到越来越多的开发者采用。越来越的的年轻人开始使用了Python3。而且由于Python目前已经发展成为最受开发者欢迎的应用程序开发和设计应用语言之一。

目前Python的优化还没有那么完善，有很多仅仅用于加快速率的补丁还没有应用到Python中去。导致现在很多人都以为Python很慢，但其实很多程序并不要求运行速度。不过也有些项目对运行时间要求会比较高，这时很多Python开发师会选用C语言来重写这些程序。

Python是一种完全面向对象的语言。它的对象包括函数、数字、字符串等，并且它具有简单方便、解释性强、可移植性等特点。

## 2.2 Django

Django是针对Python的Web开发框架，它内部存有各种Web开发组件，还具有轻量级的Web服务器。这让管理者能高效的开发和调试的各种Web应用程序[4]。

MVC是指模型（Model），视图（View），控制器（Controller）三个模块。 Django不同于传统的MVC模式，它的Model使用orm技术，比传统的sql语句操作起数据库来变得方便许多。同时动态的逻辑处理和静态页面的开发被分离开来[5]。在Django中框架有着自动处理某些用户数据的自动化功能。使用 Django 框架，可以减少很多开发中的繁琐的工作，让开发者将更多的精力专注于编写业务逻辑，而不是重复生产基础组件的工作[6]。Django 框架的核心组件有：

1.创建模型对象的映射关系；

2.为最终用户设计较好的管理界面；

3.URL设计；

4.设计者友好的模板语言；

5.缓存系统。

## 2.3 MySQL

MySQL是一个关系型数据库的操作系统,它本身具有开放式的数据库架构,能够广泛提供于用户在平台上进行多重选择,并且随着数据库技术的日渐成熟,MySQL数据库所支持的平台软件功能越来越多,性能也在不断的得到提升,同时它所支持的应用平台软件数也在日渐的增加。它的功能齐全且易于管理，同时操作人员也可以通过多种不同的语言来访问MySQL数据库[7]。

MySQL的数据管理特点主要是将不同的分类数据管理功能分别存放在多种数据库或表单里,然后再将这些存放在表单的数据存放入不同的分类数据库里,这样的数据管理做法从根本上避免了所有的数据全部分类冗杂在一起,可以更加方便快捷的利于分类使用者使用这些查询和分类的语言管理数据库,同时它也使开发数据库变得更加方便和易于分类管理。一般常用SQL语句来操作MySQL数据库。SQL语句使管理员对分类数据库或表单进行查询和存取分类信息的操作变得更加方便。它的数据管理特性主要有:

(1)主要由C和C++开发，并经过了各种编译测试，保证了代码的可移植性[8]。

(2)通过最新的SQL数据查询执行,大幅提高了数据查询的执行速度。

(3)能够直接作为一个数据库而直接嵌入到其他的数据库软件中提供多种数据库语言的支持。

(4)是TCP/IP、和jdbc等多种类型数据库之间相互关联的重要桥梁。

(5)为数据库管理提供了各种用于进行系统管理、检查、优化以及数据库相关操作的各种系统管理解决工具。

(6)系统可以同时运行处理一个同时拥有上千万条业务记录的大型业务数据库。

## 2.4 Xadmin

X-admin系统是基于一个X-layui的轻量级移动前端平台服务器和其前后台终端资源整合管理软件系统开发框架,简单安全免费,兼容性好,面向所有不同层次的前后台终端平台应用程序。它的关键技术的功能和其特性的主要表述有:

(1)基于bootstrap3:X-admin主题模板使用了bootstrap3.0框架精心的打造,这使得X-admin可以支持不同尺寸的屏幕上进行快速的浏览,同时也支持bootstrap的主题模板,让管理员管理网站后台时的样式更加动感起来。

(2)后台管理内置插件功能丰富:X-admin后台管理作为一款全面的分布式后台数据管理系统和软件框架,软件中内置了丰富的扩展管理插件和功能。

(3)强大的框架和开发系统:X-admin的插件系统设计参考了其他一些别的结构的框架的插件系统设计方法和,让我们的插件系统可以直接用来扩展系统的任何一个框架的功能和特点。

(4)强大的针对数据精确过滤器以及筛选分析功能:针对数据的精确筛选及其过滤器功能是X-admin的另一个最大强项。通过X-admin的高级数据范围筛选器和过滤器,管理员可以方便的选择使用模糊间隔时间数据范围高度查找,数字间隔时间数据范围高度查找,日期间隔时间数据范围高度查找等多种高级时间数据范围查找的解决方式。并且,X-admin还提供可以将网页查询处理结果直接将其保存为书签中方便使用者今后的查询使用。

(5)具有完善的软件系统和功能控制管理权限:X-admin的安全的软件系统,它里面的控制功能几乎可以精细到每行的数据，这使得基本上每个开发员都已经可以完全轻松胜任任何一种非常变态的系统权限管理控制和对系统的功能要求。

## 2.5 Ngnix

Nginx系统是一个高效率的web服务器，有着高效的负载均衡服务。Nginx系统采用高性能的cpu语言进行开发和编写,使它不管是在数据管理的花费上还是CPU的使用效率上都比一般服务要好得多。

Ngnix最大的特点之一就是服务器占用的空间少,并发解决能力快。实际说明使用Nginx的内存并发处理能力确实在同类型的企业网页代理服务器中总体上表现得比较好。在用户操作过程中,Nginx的配置非常便捷,启动后便能一直连续运行[9]。中国大陆市场上使用Nginx的企业网站服务器有很多,比如腾讯、百度、淘宝等等。

## 2.6 Pycharm

本次设计我使用的编程工具时Pycharm。它具有比较完善的可以用于协助开发者学习和使用Python编程语言进行开发时高效的软件和工具。同时,此IDE软件还有很多特殊的功能,比如同时能用Django框架进行各种Web软件的开发。这些高级功能使得Pycharm软件成为了Python专业开发者和新手用户都可以使用的有力解决工具。

## 2.7 Navicat

Navicat软件是一种迅捷、简单、价格实惠的中小企业数据库系统管理工具,主要是为了方便中小企业数据库的系统管理及维护和减少数据库系统管理的花销而进行设计。它的功能设计很好地符合了各类开发管理人员的各种应用需求。Navicat软件有着人性化的用户图像界面,使得管理员都可以安全快捷的查看和访问数据库信息[10]。

# 第3章 系统需求分析

## 3.1 用户相关功能需求分析

这个部分的功能主要针对学生用户的一些基础属性来展开说明。用户在第一次进入网站时，需要完成注册功能，输入相应的用户名和密码即可成功注册，然后每次输入正确的用户名和密码后即可完成登录。用户在个人中心页面可以查看个人信息，同时也可以修改个人信息。用户在个人中心页面中还能查看到已学习的课程及基本课程信息，在我的的收藏中能查看到用户收藏的课程信息。该用户功能模块需求如图3.1所示：

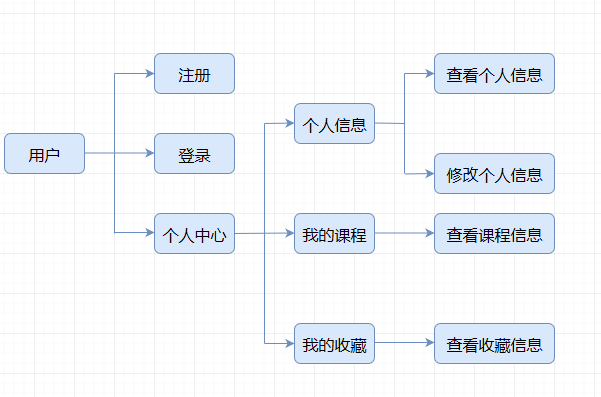


图3.1 用户功能模块

## 3.2 云课堂功能需求分析

云课堂上提供的视频服务主要面向的对象为学生用户，学生能在首页对平台内所有的课程进行全局搜索，同时能按更新时间查看到最新的课程列表，如果学生对其他热门课程感兴趣，平台内也提供了热门课程推荐的功能。进入课程详情后学生用户能收藏课程，或者直接进行视频的学习，在视频播放页用户能下载课程的相关资源，同时还能查看到老师的一些课前的温馨提示,具体的相关功能需求如图3.2所表示：

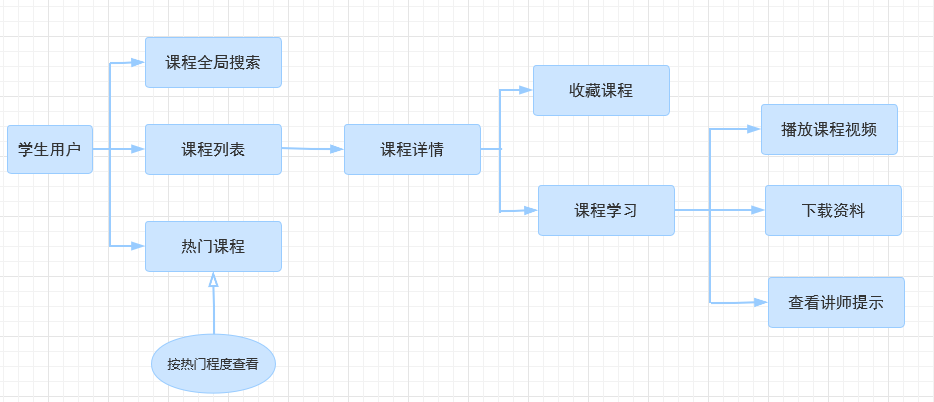


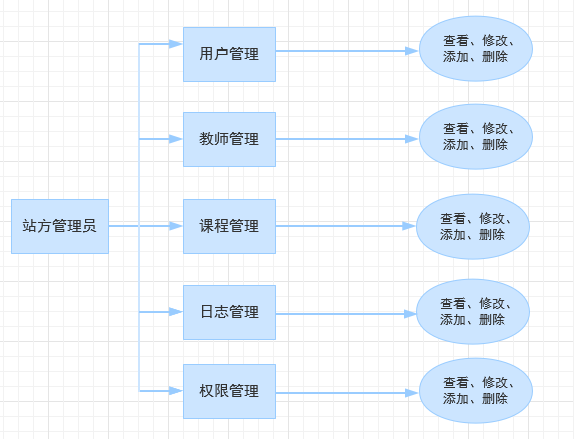
图3.2 云课堂功能模块

## 3.3 后台管理功能需求分析

后台管理模块主要是让管理员来维护网站的的一些功能，是为网站提供一些数据信息的主要途径。本次设计我们把后台管理模块主要分为五个部分，各个部分各司其职互不干扰，有效的提高了网站运维的效率，具体的功能分析流程图如图3.3。具体用例分析如下：

1. 用户管理。站方可以对用户的用户信息、用户的课程收藏等进行管理，可以对以上信息进行查找、修改、添加、删除。
2. 教师管理。站方可以对教师的个人资料、教师所授的课等进行管理，可以对以上教师的基本信息进行查询、添加和删改。
3. 课程管理。管理员方可以对课程的章节、资源、详情介绍等一些基础信息进行管理，可以对任意课程进行查找、修改、添加、删除。
4. 日志管理。站方在后台可以查看到任意用户在后台的一些改动信息日志。
5. 权限管理。站方可以为不同的模块分组设置不同的权限。

图3.3 后台管理功能模块



# 系统设计

## 4.1 系统总体设计

本次设计我们整个云课堂平台系统采用的是B/S架构，系统整体的框架设计如图4.1所示：

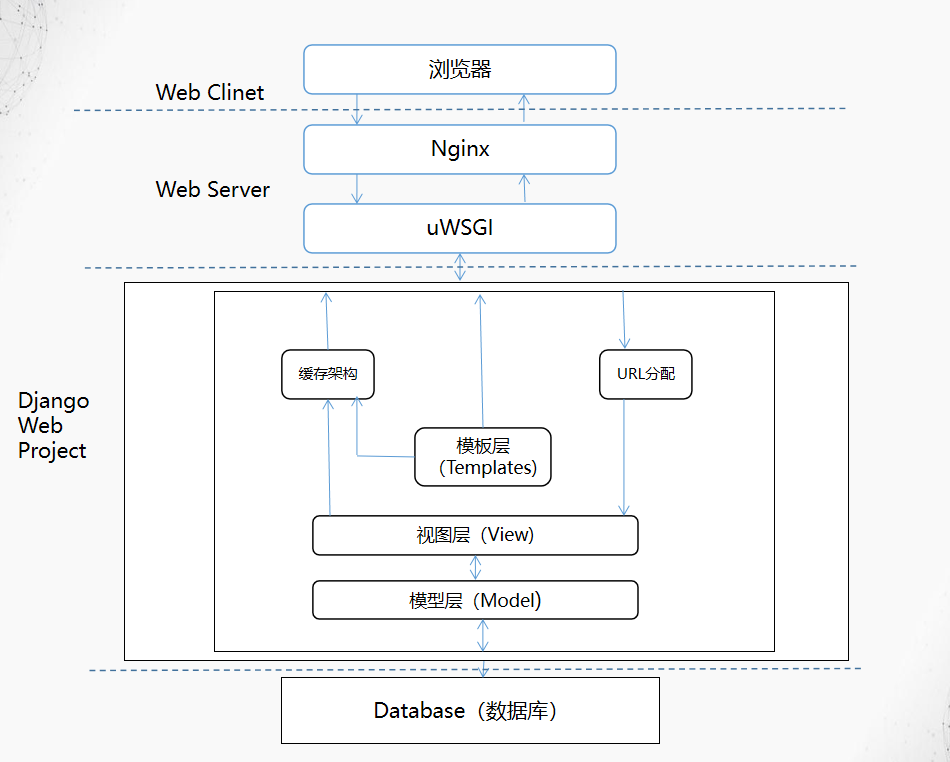


图4.1 系统总体设计框架

从图中我们可以看出整个系统被划分成了三个部分，分别是Web服务器端、Django的Web项目和数据库这三大部分。同时对这三大部分作出分析如下：

（1）服务器。为了实现Web服务，并提供更高的效率和负载均衡的要求，使用Nginx和uWSGI作为本次设计的Web服务器。Nginx主要用来处理静态的HTML页面，动态请求则主要由uWSGI处理，它会根据uwsgi协议把请求转运到Django处，再由Django内部处理转来的相应的业务逻辑[11]。

(2)框架。Django作为云课堂平台的核心Web框架,主要用来处理Web相关的业务逻辑和高速缓存架构的开发[9]。总体框架分为三个层次,即所谓MTV模式:

第一层,Model（模型层）,即关系型数据库的存取层,需要在该存取层完成关系型的数据库(Mysql)与数据库业务实体数据库对象的模型定义与关系,具体每个模型的类都与一张数据库表类型对应。

第二层,View(视图层),也为处理项目的逻辑层,主要用来接收URL请求并反馈回应。View层使用基于类的视图来操作视图层,用类方法处理http方法中的问题,同时也具有各种可扩展需求[11]。

第三层,Template(模板层),专门用于表现内容的一层,主要用来显示网站的前端页面。概括来说就是如何往 HTML 文件中填入动态内容的系统。

（3）数据库。这次设计我使用的数据库是关系型数据库MySQL,MySQL使用的是C/S体系结构，所以实现功能时会分为两个程序：一个是能取出数据的MySQL服务器端程序，它能够接收客户端的请求并通过相应的请求去查找数据库的内容，再把得到的回应传回给客户；还有一个能存储数据的是MySQL客户端程序，它负责连上服务器，通过发出请求来传递给服务器它所需要的操作。

通过以上构架的设计模型以及服务器、web框架和不同的功能组件、数据的分析和组合等进行设计,可以将其作为我们设计不同功能模块的工具和基础,同时也能有效达成我们构建中国大学生高等教育云课堂信息服务平台的战略目标。

## 4.2系统数据库设计

### 4.2.1数据库设计概述

数据库是整个系统的基石,一个完整系统的成败与否在于其数据库的设计。本节将对本平台需要进行的数据库的设计进行讲述[12]。

云课堂平台，即网上教学平台，由于它不一定是实时的教学方式，所以它需要一个稳定的服务器，用户则需要一个能连网的终端机。同时为了方便对大量数据的增删改查，最好还须有一个易于管理的数据库[13]。数据库管理是用于进行数据系统管理的最新应用技术。近几年来数据库的管理系统已经从个人私用的数据库应用程序渐渐的变成为现在普通使用的信息系统管理软件。大型数据库管理系统有着数据结构多元化,冗余度低,程序与应用数据的独立性相对比较高等优势，所以目前很多大型的信息库管理系统都有建立在数据库管理设计之上。

数据库系统的出现进一步让计算机信息管理系统从以加工和处理数据的应用程序数据库为主的管理中心从而逐步转向了一个环绕数据共享的以信息数据库系统作为管理中心的新发展阶段。这样一来即可以有利于充分方便了应用程序的研发和运行维护,又可以有利于实现数据的自动化和集中管理。目前,数据库已经是现代计算机信息管理系统中一个不可缺少的组成部分。因此数据库的技术也进一步成为了当今的计算机信息管理领域中技术发展速度最快的计算机技术之一。

数据库的设计过程是建立在一个数据库及其相关应用的系统设计基础之上的一个核心设计问题。一个数据库及其相关应用的整体功能都是基于一个比较完善的数据库应用系统设计的理念和基础之上,数据库的模型和数据信息是一切设计操作的依据和基础,如果一个数据库应用系统设计的不好,那么其它一切可以用于改善和提高其数据库应用性能的设计方法都可能是有限的。所以数据库应用系统设计的关键组成部分主要内容在于如何能够使用户自己设计的一个数据库系统能够合理地收集和存储用户的信息和数据,方便于用户对这些数据信息进行正确的处理。

设计关系型技术数据库必须严格地要求遵循设计的基本原则,在MySQL的设计数据库中,这种遵循设计基本规则的理论基础就是一种设计规则范式,范式一般指的是能够符合某一种特定设计级别的数据关系处理模式的设计规则数据集合。对数据库采用第三方规范式技术减少了大量系统数据冗余,节约了传统数据库文件存储的大量空间,同时大大程度加快了其在数据库中的增、删、改的日常工作处理速度。

### 4.2.2 系统数据表结构设计

数据库的设计与系统功能需求相辅相成，一个合理的数据库设计可以有效提高系统的效率和可扩展性。数据库用于应用程序服务，首先要满足系统需求，合理处理表之间的关联关系，确保数据没有过多的冗余和完整性，确保数据表的可扩展性以应对新的需求[14]。

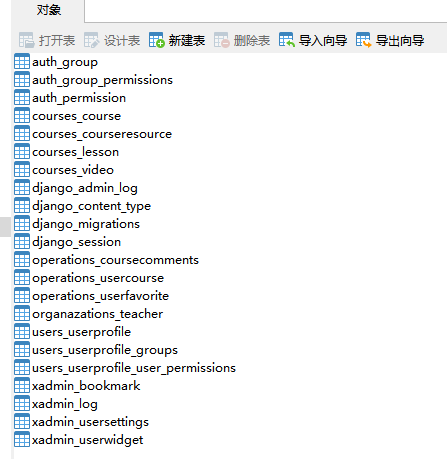
首先在本项目中的settings.py文件中建立项目与数据库的连接代码如下：

DATABASES = {  
 'default': {  
 'ENGINE': 'django.db.backends.mysql',  
 'NAME': "xuexi",  
 'USER': "root",  
 'PASSWORD': "lu1998529",  
 'HOST': "127.0.0.1",  
 }  
}

云课程平台的数据库的设计主要集中在用户信息，教师信息和课程信息等各部分。以下是系统中主要实体及属性的定义：

1. 管理员：管理员用户名、管理员密码、最后登陆日期。
2. 学生用户：性别、昵称、生日日期、头像。
3. 教师：姓名、头像、所授课程
4. 课程：课程所属授课老师、课程名、课程描述、学习时长（分钟数）、课程类别、课程标签、课程详情、课程须知、老师忠告、课程点击数、课程封面图、创建课程时间。
5. 章节：章节所属课程、章节名、学习时长（分钟数）。
6. 视频：视频所属章节、视频名、学习时长（分钟数）、访问地址。
7. 课程资源：课程资源所属课程、课程资源名、下载地址。

然后在django项目的各个模块的models.py中设计各个数据表，最终设计的表单结构如下：

 图4.2表单结构

同时我们对用户信息表也进行了相应的设计，用户信息表的设计如表4.1所示：

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名 | 字段类型 | 属性 | 描述 |
| id | int | 主键 | 用户ID |
| password | varchar | 非空 | 密码 |
| last\_login | datatime | 非空 | 上次登录时间 |
| is\_superuser | tinyint | 非空 | 是否为管理员 |
| username | varchar | 非空 | 用户名 |
| first\_name | varchar | 非空 | 英文名名字 |
| last\_name | varchar | 默认为空 | 英文名姓氏 |
| email | varchar | 非空 | 邮箱 |
| is\_staff | tinyint | 非空 | 是否为员工 |
| is\_active | tinyint | 非空 | 是否激活 |
| date\_joined | datatime | 非空 | 加入日期 |
| nick\_name | varchar | 非空 | 昵称 |
| birthday | data | 默认为空 | 生日 |
| gender | varchar | 非空 | 性别 |
| image | varchar | 非空 | 头像 |

表4.1 用户信息表

课程详情表的设计如表4.2所示：

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名 | 字段类型 | 属性 | 描述 |
| id | int | 主键 | 课程ID |
| name | varchar | 非空 | 课程名 |
| desc | longtext | 非空 | 课程描述 |
| learn\_times | int | 非空 | 学习时长 |
| category | varchar | 非空 | 课程种类 |
| tag | varchar | 非空 | 课程标签 |
| add\_time | datatime | 非空 | 添加时间 |
| detail | longtext | 非空 | 课程详情 |
| image | varchar | 默认为空 | 封面图 |
| teacher\_id | int | 非空 | 老师ID |
| click\_nums | int | 非空 | 点击数 |
| teacher\_tell | longtext | 非空 | 老师公告 |
| youneed\_know | longtext | 非空 | 课程须知 |

表4.2 课程详情表

课程章节信息表的设计如表4.3所示：

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名 | 字段类型 | 属性 | 描述 |
| id | int | 主键 | 章节信息ID |
| add\_time | datatime | 非空 | 添加时间 |
| name | varchar | 非空 | 章节名 |
| learn\_times | int | 默认为空 | 学习时长 |
| course\_id | varchar | 外键 | 课程ID |

表4.3 课程章节信息表

视频信息表的设计如表4.4所示：

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名 | 字段类型 | 属性 | 描述 |
| id | int | 主键 | 视频信息ID |
| add\_time | datatime | 非空 | 添加时间 |
| name | varchar | 非空 | 视频名 |
| learn\_times | int | 默认为空 | 学习时长 |
| url | int | 非空 | 视频地址 |
| lesson\_id | varchar | 外键 | 章节ID |

表4.4 视频信息表

|  |  |  |  |
| --- | --- | --- | --- |
| 字段名 | 字段类型 | 属性 | 描述 |
| id | int | 主键 | 资源ID |
| add\_time | datatime | 非空 | 添加时间 |
| name | varchar | 非空 | 名称 |
| file | file | 非空 | 资源文件 |
| course\_id | varchar | 外键 | 课程ID |

课程资源表的设计如表4.5所示：

表4.5 课程资源表

## 4.3系统功能模块设计

### 4.3.1用户模块设计

用户信息管理模块的基本功能设计主要包含有用户登陆、注册、查看以及修改用户个人信息这些模块的基本功能,以下主要是对于用户信息管理模块的这些基本功能的一个详细的设计及其说明:

1. 注册登录

用户登录首先进入我们的注册会员登录页面,然后直接点击输入自己想用的账号以及密码即可直接完成用户注册；同时只要用户在登陆的页面,输入自己注册过的账号以及密码即可直接完成用户登录。我们首先就需要用一张用user的表格格式来记录和保存注册用户的信息,基本的设计代码如图4.6所示:

图4.6 user表模块代码

其中定义的：

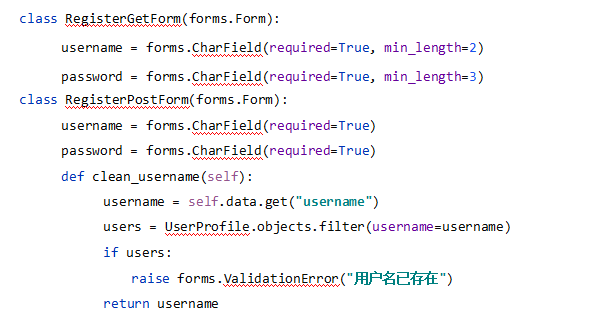
nicheng：必填，最长不超过50个字符，并且唯一，不同名

nima：必填，最长不超过50个字符

xingbie:必填，使用了一个choice，只能选择男/女

数据中定义是按照用户创建时间的排列。

同时django在内部开发中还有一个能快速处理表单的功能,它用面向对象的方法,通过利用python语言生成前端代码,专门用于帮助我们快速获取和处理与表单代码相关的数据和内容。编写表单django的过程生成form表单,就相当于我们在自己的模型管理系统里简单地编写了一个允许用户登陆或者注册的表单模型,编写的过程如图4.7所示：

图4.7 登录注册表单设计

（2）个人中心查看

用户在完成了平台登录或注册相关操作后后即可直接通过右上角点击个人中心即可进入用户信息管理中心,通过左侧的导航切换即可直接查看或修改用户的个人信息和个人课程收藏。

在用户的个人信息中,用户在初次通过系统进入时可以直接看到一个系统设定的用户默认昵称和头像。同时用户需要更新和完善自己的一些其他的个人基础信息,其中有用户的昵称、生日日期、性别和默认头像。

在我的课程收藏里,用户第一次查看课程收藏进入时显示为空的收藏页。如果用户已经产生过第一次收藏的行为,则该用户便可以在此列表中找到已经收藏过的所有课程。

（3）个人中心修改

用户在完善个人信息后,可随时对个人信息进行修改。除了头像外其他信息都可以直接在对应的表单中输入新的信息,然后点击保存就会有提示修改成功的标志并将修改后的信息呈现在表单中。如果要修改头像，就需要重新上传图片然后点击保存。

对于个人信息功能模块来说需要处理用户在云课堂中是否进行了登录的问题后我们才能直接进入并查看或修改。,所以我们需要定义一个多重继承的类来实现用户的一些其他功能。Python视图中需要使用这种多重继承的方法,因此我们需要在视图中定义了一个父类,主要用于帮助系统检测用户的登录状态,如果不是登录状态系统就会自动回到用户的登陆页面。

### 4.3.2云课堂模块设计

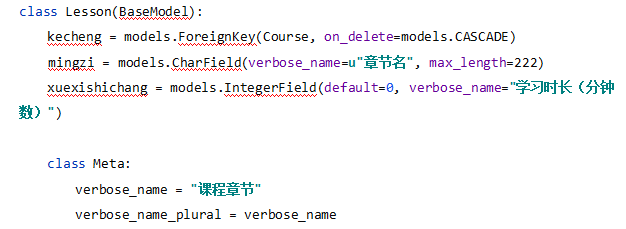
云课堂模块是整个系统模块中最核心的模块之一，该模块主要为学生用户提供服务。在该模块中我们定义了Course、Lesson、Video、CourseResource这四大类，其中

Course类的定义的代码如图4.8所示：

图4.8 Course类的定义

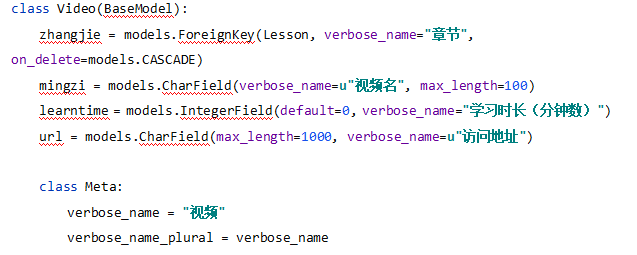
在课程模块中click\_nums，即用户对于课程的点击量，将会作为热门课程推荐按热门程度排序出现于课程列表页的右端中。用户在同一个课程的列表页中点击一次课程进入课程detail列表页面时,便记为一次课程的点击量。用户对相同课程的点击量可以进行重复记录。

Lesson类的定义如图4.9所示：

图4.9 Lesson类的定义

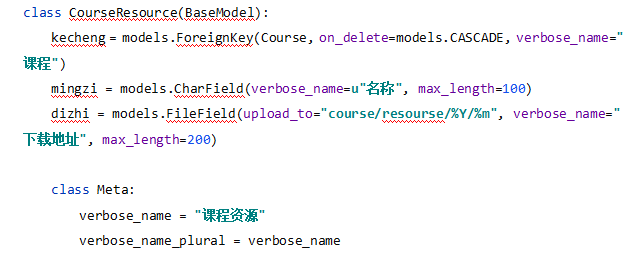
在课程章节模块中我们给每个课程定义了不同的章节名，同时在进入课程详情页时用户能对课程进行收藏操作，该操作允许逆向进行，如用户点击已收藏则可取消收藏，系统将会删除此条行为记录。

Video类的定义如图4.10所示：

图4.10 Video类的定义

在视频模块中我们给每个章节设定了不同的视频信息，通过管理员设置的url学生用户能看到视频的播放。

CourseResouce的定义如图4.11所示：

图4.11 CourseResouce的定义

在课程资源模块中我们给每个课程提供相对应的课程资料，每个课程资料都配有相应的下载地址，学生用户可点击下载地址下载课程资源。

### 4.3.3后台管理模块设计

后台的管理模块是用来为后台管理员以及用户系统提供一些平台的基础数据、数据的收集和管理等,由于Django本身自带的admin管理模块具有一定的功能限制,本平台以及用户系统我们需要采用第三方的Xadmin来设计后台框架。它会从django内部定义的Model中,自动的生成一个基础信息和数据自动管理的页面,从而实现数据的查找等操作,也为后台提供了比较强的功能和扩展性。

根据图3.3所述的五种用例，用户、教师、课程、日志、权限每一种类型的用例都可以根据需求构建相对应的模型model,其中对于用户、教师、课程的权限模型已在本章关于数据库系统设计的部分中自动完成了设计,而对于日志和课程权限的自动控制模型,则已经被xadjango自动地创建。根据Xadmin的特性将后台管理模块统一分为两类：一类是数据的显示页面，及列表页，通过这个页面可以在里面完成查询、过滤等功能。另一类是具体的数据操作页面，包括增加、修改、删除等功能，通过这个页面我们可以完成对数据的具体操作。

在pycharm中添加Xadmin的方式很简单，只需要在github上下载相应xadmin源码，然后在我们项目settings的INSTALLED\_APPS中添加crispy\_forms 和 xadmin，然后再安装xadmin的一些依赖包，最后通过migrate生成xadmin需要的表，即可使用xadmin这个功能。

# 第5章 系统实现

## 5.1系统总体框架实现

### 5.1.1系统开发环境

本次设计的云课堂平台主要是在win10的操作系统下完成的。该平台基于Django框架进行开发，用到的python版本为3.7。对于数据库我们主要采用的是Mysql关系数据库。对于后台的开发主要用到的管理框架是Xadmin。其他的主要的开发工具和系统使用环境如图5.1和图5.2所示：

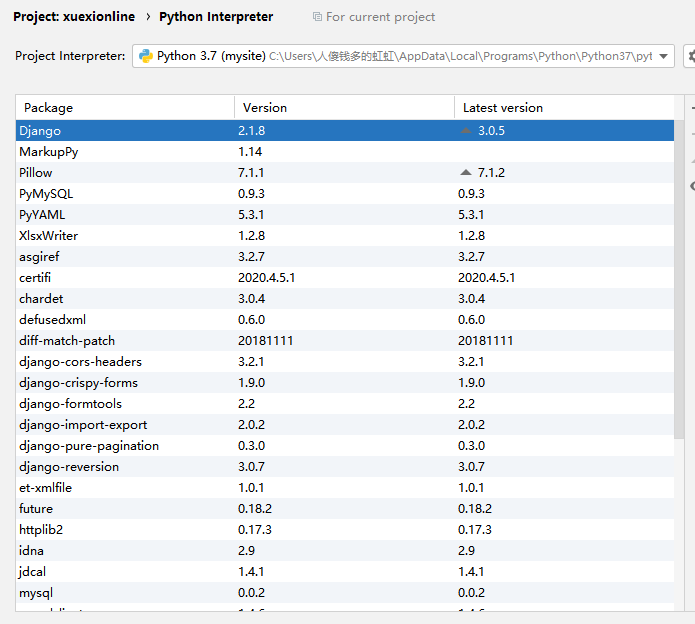
 图5.2 主要开发环境和工具

图5.1 主要开发环境和工具

### 5.1.2项目组织结构





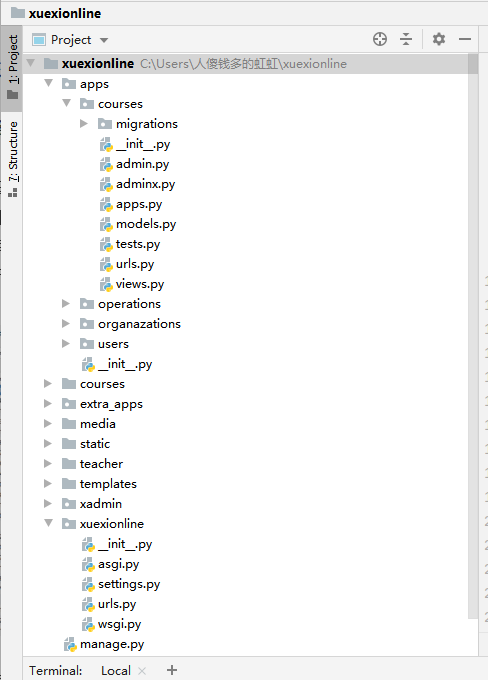


图5.3 项目组织结构图

由图可见总体项目结构主要分为xuexionline、apps、media、xadmin、static、templates、manage.py这几个文件和文件夹来实现相应的功能。

xuexionline，即项目的主文件夹。在它的settings中能注册到项目所有的app,同时在里面可以进行设置database，这样能直接完成项目和数据库的连接。其中还有urls.py文件,它的主要作用就是在对url的请求和负责处理该项目请求的任何一个视图函数之间很好地建立一个互相对应的关系,也可以说它就是一个对url请求的映射表。它的具体处理工作和流程具体说明如下:1.客户端向浏览器先发送一个请求2.浏览器的服务端会接受这个url,并在一个项目的所有视图应用(其中包括根目录)的一个urls.py配置文件中对请求进行了查找,如果这个url能被找到,服务器端就会将该url请求提交给其相对应的视图函数来对其进行管理。3.视图函数处理完url后会把这些业务数据,通过发送到returnrender(request,'模板文件',数据);然后把文件渲染到前端页面便能显示出来。

apps,主要用于处理项目的核心业务。在apps文件夹下我们又建了四个子文件夹courses、operations、organazations、users，分别用来实现云课堂平台中的课程、教师和用户功能，各个部分相互独立处理，使得管理员在管理整个项目中变得更为便捷。

media,用于存储用户上传的文件。

static，用于存储css、js和一些图片文件。本地开发时用django处理，部署时需要使用Nginx处理。

templates，用来存放项目的html模板文件。

manage.py,用于执行操作命令的python文件。

## 5.2 用户模块实现

### 5.2.1 登陆注册

（1）注册

实现平台注册功能一般采用view继承的方式,首先定义一个可以实现注册功能的RegisterView类。在RegisterView下采用get()方法可以直接返回注册页面。在Django中的urls.py下进行路由的配置path("register/",RegisterView.as\_view(),name=“register”)。在urls中,采用类的as\_view（）方法,然后调用这个View类。之后还要进行模板的修改，为了能顺利跳转到注册页面需要修改index.html,此时点击“注册”便能跳转到用户注册页面。同时改动register.html中的静态文件地址,测试从index界面点击注册时能否跳转到注册界面。完成相关配置之后,进行RegisterView后台逻辑的编写,在该类中添加get方法,利用get方法实例了一个form表单,并通过render谊染html将页面返回给用户[15]。在类中添加post方法,如果是post请求,先生成一个表单实例,并获取用户提交的所有信息(request.POST)。使用is\_valid()方法来验证用户的提交信息是否合法。如果用户已经存在,则将msg提示的“用户己存在”返回页面[15]。如果合法,将存放用户提交的password。将user\_profile对象实例化,此时它能把用户提交的个人注册信息添加到数据库。根据msg信息,有用户名或密码长度少于两位或用户名重复等错误提示信息。实现如图5.4所示，类的代码如下：

class RegisterView(View):  
 def get(self, request, \*args, \*\*kwargs):  
 register\_get\_form = RegisterGetForm()  
 return render(request, **"register.html"**, {  
 **"register\_get\_form"**:register\_get\_form  
 })  
  
 def post(self, request, \*args, \*\*kwargs):  
 register\_post\_form = RegisterPostForm(request.POST)  
 if register\_post\_form.is\_valid():  
 username = register\_post\_form.cleaned\_data[**"username"**]  
 password = register\_post\_form.cleaned\_data[**"password"**]  
 *# 新建一个用户* user = UserProfile(username=username)  
 user.set\_password(password)  
 user.username = username  
 user.save()  
 login(request, user)  
 return HttpResponseRedirect(reverse(**"index"**))  
 else:  
 register\_get\_form = RegisterGetForm()  
 return render(request, **"register.html"**, {  
 **"register\_get\_form"**:register\_get\_form,  
 **"register\_post\_form"**: register\_post\_form  
 })

图5.4 注册页面

1. 登录

用户从网站首页进入登录界面后,需要在登录框中填写正确的用户名和密码才能完成登录。其中authenticate()方法用来验证在数据库中已经加密后的密码,如果验证成功,便可以成功返回user对象。可以利用form表单的形式来设计登录表单的验证,定义好form表单后,即可以使用它来做验证,并能完善错误信息。根据msg信息,系统设置了提示用户名或密码错误的提示信息。

本次平台的登录模块也和注册模块一样采用的是view类的继承方式,通过view类的继承,可以实现负责登录功能的LoginView。实现如图5.6所示，类的代码如图5.5所示：

图5.5 LoginView类的定义

![BQ[{[$UX`72{E]U](5%EM7F](data:image/png;base64,)

图5.6 登陆页面

### 5.2.2 个人中心

（1）个人信息

实现个人信息功能也是采用view继承的方式,通过定义相关的视图类来完成个人信息的查看及修改功能,主要定义了两类：UserInfoView和UploadImageView。UserInfoView中的get方法可以用于回到个人主页。在用is\_valid()验证后，利用sava（）方法将数据保存到数据库。UploadImageView主要用来实现用户上传头像，同时也需要实现post方法。在进入个人中心前用上了LoginRequiredMixin方法来检测用户是否已经登录，如果没有进行登录的话便会跳转至登陆页面。实现如图5.7所示，类的代码如下：

class UserInfoView(LoginRequiredMixin, View):  
 login\_url = **"/login/"** def get(self, request, \*args, \*\*kwargs):  
 current\_page = **"info"**  return render(request, **"usercenter-info.html"**,{  
 })  
 def post(self, request, \*args, \*\*kwargs):  
 user\_info\_form = UserInfoForm(request.POST, instance=request.user)  
 if user\_info\_form.is\_valid():  
 user\_info\_form.save()  
 return JsonResponse({  
 **"status"**:**"success"** })  
 else:  
 return JsonResponse(user\_info\_form.errors)

class UploadImageView(LoginRequiredMixin, View):  
 login\_url = **"/login/"**def post(self, request, \*args, \*\*kwargs):  
 *#处理用户上传的头像* image\_form = UploadImageForm(request.POST, request.FILES, instance=request.user)  
 if image\_form.is\_valid():  
 image\_form.save()  
 return JsonResponse({  
 **"status"**:**"success"** })  
 else:  
 return JsonResponse({  
 **"status"**: **"fail"** })

 图5.7 个人信息

（2）我的收藏

用户在进入个人中心后在我的收藏里可以看到收藏的课程和授课教师，这两类业务分别通过MyFavCourseView和MyFavTeacherView两个类来实现。MyFavCourseView的实现流程是View中的get方法在接收到用户需要查看收藏的请求后，为了实现现实页面的逻辑，首先定义了一个course\_list=[]的空的课程列表，之后通过filter()可以查询之前定义的UserFavorite的表中用户对于课程的收藏状况。fav\_id表示不同的收藏类型，在本平台中定义的fav\_id为1代表的是收藏课程，fav\_id值为2代表的是收藏的授课讲师。最后定义一个render返回课程列表，并为它添上对应的url路由配置：url(**r'^myfav\_course/$'**, MyFavCourseView.as\_view(), name=**"myfav\_course"**)。MyFavTeacherView类的逻辑也同上。实现如图5.8所示，类的代码如下：

class MyFavCourseView(LoginRequiredMixin, View):  
 login\_url = **"/login/"** def get(self, request, \*args, \*\*kwargs):  
 current\_page = **"myfav\_course"** course\_list = []  
 fav\_courses = UserFavorite.objects.filter(user=request.user, fav\_type=1)  
 for fav\_course in fav\_courses:  
 try:  
 course = Course.objects.get(id=fav\_course.fav\_id)  
 course\_list.append(course)  
 except Course.DoesNotExist as e:  
 pass  
 return render(request, **"usercenter-fav-course.html"**,{  
 **"course\_list"**:course\_list,  
 **"current\_page"**:current\_page  
 })  
  
class MyFavTeacherView(LoginRequiredMixin, View):  
 login\_url = **"/login/"** def get(self, request, \*args, \*\*kwargs):  
 current\_page = **"myfav\_teacher"** teacher\_list = []  
 fav\_teachers = UserFavorite.objects.filter(user=request.user, fav\_type=3)  
 for fav\_teacher in fav\_teachers:  
 org = Teacher.objects.get(id=fav\_teacher.fav\_id)  
 teacher\_list.append(org)  
 return render(request, **"usercenter-fav-teacher.html"**,{  
 **"teacher\_list"**:teacher\_list,  
 **"current\_page"**:current\_page  
 })



图5.8 我的收藏

## 5.3 云课堂模块实现

### 5.3.1 课程列表

在进入课程列表页时，用户可以在右上角通过全局搜索的方式,按照课程名或是教师名来搜索相应的课程,此时后台根据用户输入后的数据反馈到数据库中，数据库会执行select语句来查询相关课程和老师名。在经过后台的筛选后后台会将结果反馈到前端页面，如果搜索后发现没有匹配的结果，则会显示空白,并且会出现提示信息提醒用户教师名未存在或未查找到相应课程。若存在此老师或课程,则会在前端页面中显示出来。

在课程列表的首页,课程是按照后台设置的增添时间add\_time的先后顺序进行的排序，最新的课程排在前。在课程列表的右边是按照课程的点击量click\_nums进行的排序，点击最高的课程在最前，同时设置只显示点击量最高的三门课程。实现如图5.9所示，类的代码如下：

class CourseListView(View):  
 def get(self, request, \*args, \*\*kwargs):all\_courses = Course.objects.order\_by(**"-add\_time"**)  
 hot\_courses = Course.objects.order\_by(**"-click\_nums"**)[:3]  
  
 return render(request, **"course-list.html"**, {**"all\_courses"**:all\_courses, **"hot\_courses"**:hot\_courses})



图5.9 课程列表页

### 5.3.2 课程详情

用户可以通过课程列表的课程标志和热门课程推荐进入到课程详情页面。每当用户点击进入到课程详情页一次,后台数据中课程的click\_nums会加1,并保存到数据库。课程详情页中包括课程名、课程描述、课程章节数、课程时长、课程类别这些信息。课程列表页的下方有介绍课程的课程详情，这些数据都可以通过后台管理系统Xadmin来作更改。同时课程列表页显示还有两个操作按钮，分别是收藏和开始学习按钮，用户点击收按钮即可收藏成功，此时在用户的个人中心中可以查看用户收藏的课程，如果要取消收藏该课程的话再此点击收藏按钮即可。点击开始学习便会进入到课程的视频学习页面。实现如图5.10所示，类的代码如下：

class CourseDetailView(View):  
 def get(self, request, course\_id, \*args, \*\*kwargs):course = Course.objects.get(id=int(course\_id))  
 course.click\_nums += 1  
 course.save()  
 return render(request, **"course-detail.html"**, {**"course"**: course})has\_fav = False  
 if request.user.is\_authenticated:  
 if UserFavorite.objects.filter(user=request.user, fav\_id=course\_id, fav\_type=1):  
 has\_fav = True  
  
 return render(request, **"course-detail.html"**, {**"course"**:course, **"has\_fav"**:has\_fav})

**** 图5.10 课程详情

### 5.3.3 课程学习

用户从课程详情页点击开始学习即可进入课程学习页面，课程学习页主要包含每门课程的所有章节信息、视频资源、课程相关资源下载和讲师提示。在用户点击资源下载时用户应与课程相关联，此时给出一个LoginRequiredMixin逻辑进行判断用户是否登录，未登录的话系统就会自动跳转至登陆页面。用户在课程学习页能看到所有的章节显示，每个章节下有对应的子章节，点击子章节后用户便进入到了视频播放页面。本次设计的视频播放功能我们使用阿里云的对象存储服务，在阿里云的OSS控制台我们可以在里面添加任意的本地视频，然后再将其中生成的url地址添加到后台管理系统的视频地址中去，这样视频就可以显示在播放页面上了。本次要实现的继承的View有 VideoView和CourseLessonView，实现如图5.12和图5.13所示，代码实现如图5.11所示:

图5.11 Video和CourseLesson类的定义

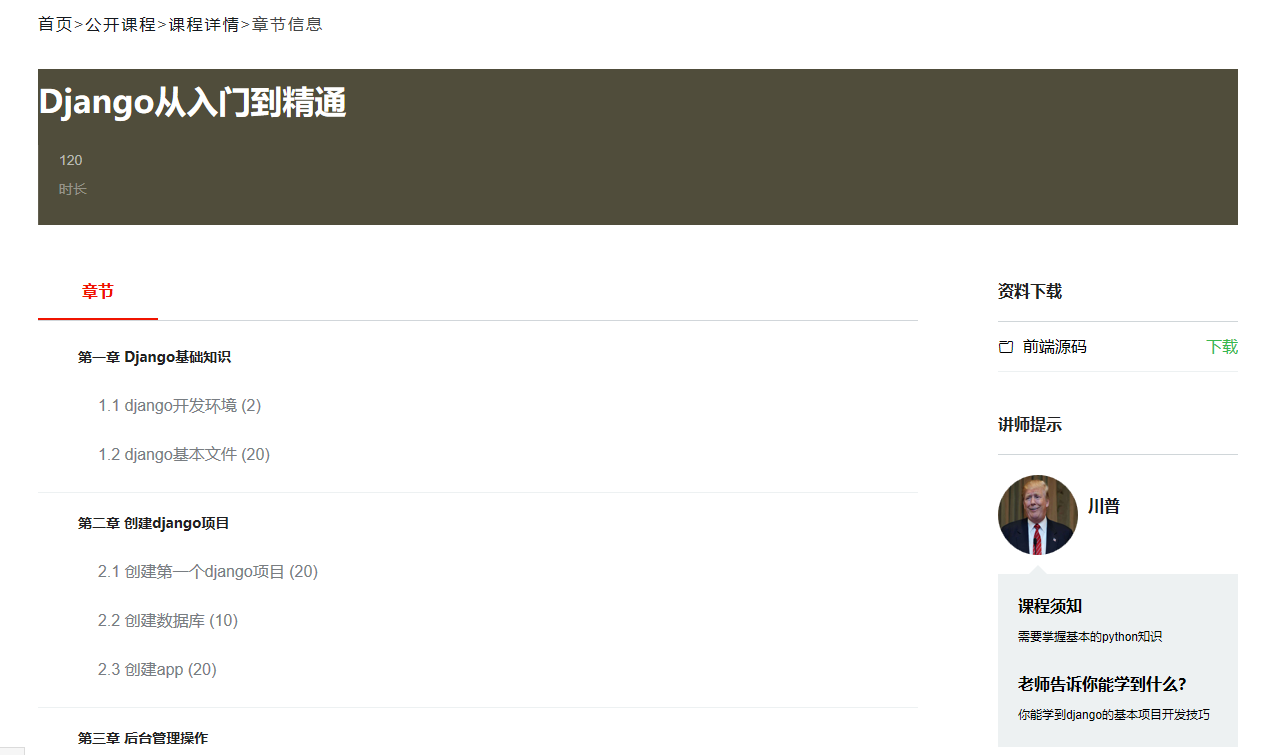


图5.12 课程学习页

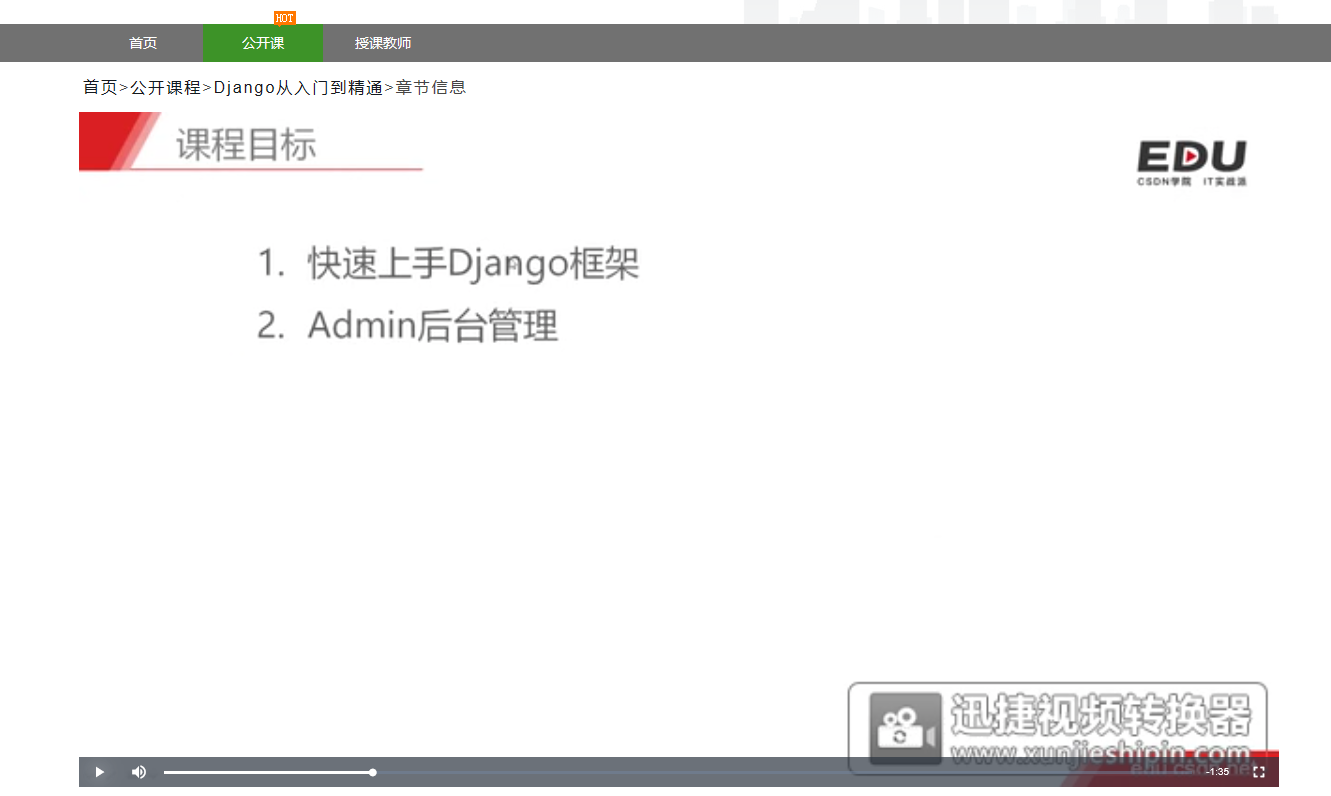


图5.13 视频播放页

## 5.4 讲师模块实现

讲师模块包括了实现讲师列表页和讲师详情页。讲师的列表页TeacherListView来完成，学生用户在进入讲师列表页时能看到讲师根据热门程度的排序，用hot\_teachers = Teacher.objects.all().order\_by(**"-click\_nums"**)[:3]命令在页面中显示了热门程度前三名的老师。讲师的详情页由TeacherDetailView定义，学生进入讲师详情页面后，能看到讲师所教的全部课程，同时点击课程就能跳转到课程详情页。实现如图5.14和图5.15所示，代码实现如下:

class TeacherDetailView(View):  
 def get(self, request, teacher\_id, \*args, \*\*kwargs):  
 teacher = Teacher.objects.get(id=int(teacher\_id))  
 teacher\_fav = False  
 if request.user.is\_authenticated:  
 if UserFavorite.objects.filter(user=request.user, fav\_type=2, fav\_id=teacher.id):  
 teacher\_fav = True   
 return render(request, **"teacher-detail.html"**, {  
 **"teacher"**:teacher,  
 **"teacher\_fav"**:teacher\_fav,  
 })  
class TeacherListView(View):  
 def get(self, request, \*args, \*\*kwargs):  
 all\_teachers = Teacher.objects.all()  
 teacher\_nums = all\_teachers.count()  
 all\_teachers = all\_teachers.order\_by(**"-click\_nums"**)[:3]  
 return render(request, **"teachers-list.html"**, {  
 **"teachers"**:teachers,  
 **"teacher\_nums"**:teacher\_nums,  
 })



图5.14 讲师列表页

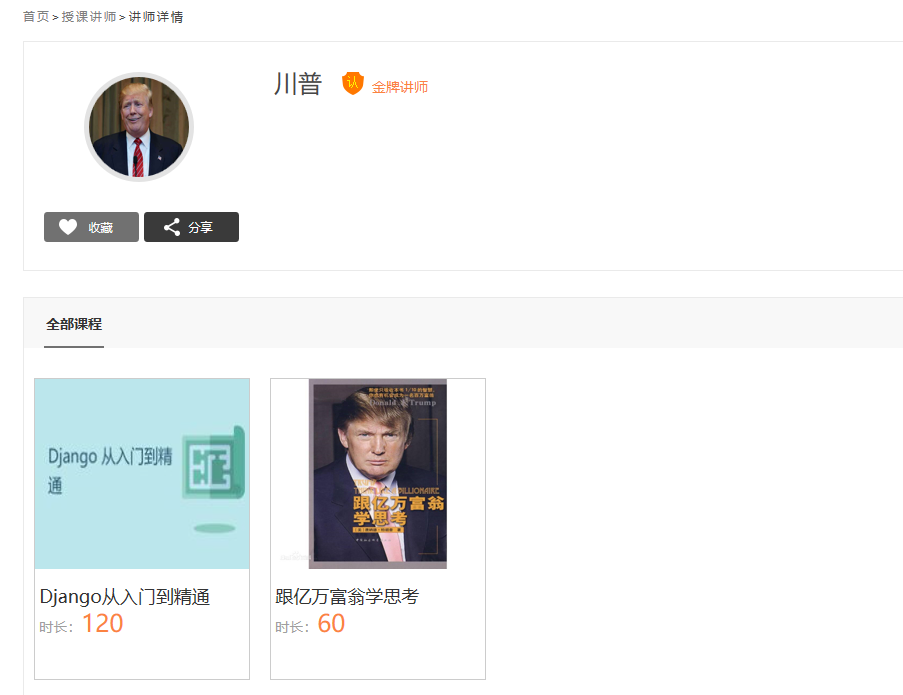
****

图5.15 讲师详情页

## 5.5 后台管理模块实现

后台管理模块主要用的是Xadmin框架，包括了讲师、用户、课程、权限、日志的管理。在每个项目app的下的模块中需要建了adminx.py文件，这样才能使xadmin中的模型管理类和各个项目模块中的模型联系起来。在这里我们主要举一个课程管理的例子来说明如何设计一个adminx.py文件。首先在一个adminx.py文件中需要建立一个名为courseadmin的管理模块类,它们基本上继承了object,并能通过list\_display文件来为该管理模块在后台的管理模型系统中进行命名。通过search\_fields能实现后台的搜索功能，list\_filter则是用来指定后台搜索时可以使用的过滤工具。最后通过xadmin.site.register(Course,CourseAdmin)命令将两类关联起来。实现如图5.16所示，代码实现如下:

class CourseAdmin(object):  
 list\_display = [**'name'**, **'desc'**, **'detail'**, **'learn\_times'**]  
 search\_fields = [**'name'**, **'desc'**, **'detail'**]  
 list\_filter = [**'name'**, **'teacher\_\_name'**, **'desc'**, **'detail'**, **'learn\_times'**]

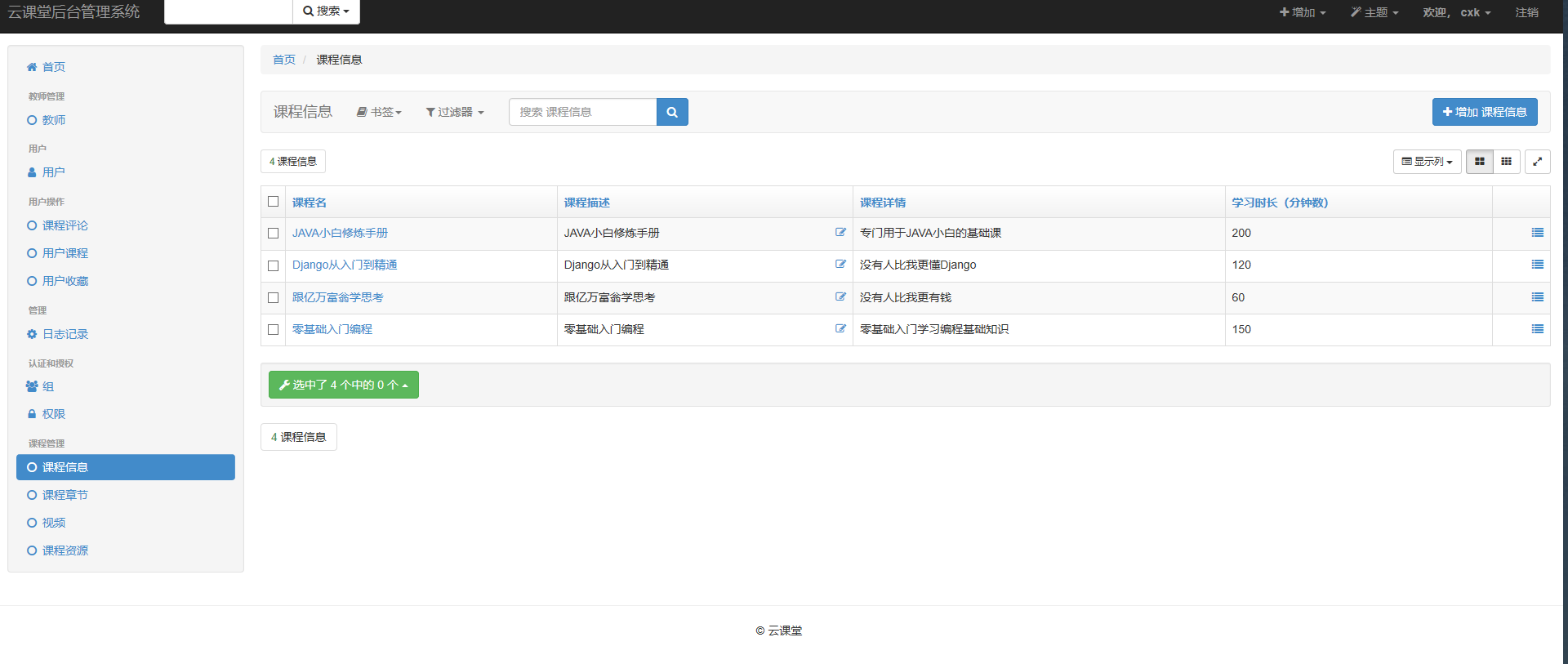


图5.16 后台课程管理页

# **总结**

本次课程设计让我第一次独立完成了一个网站的设计。虽然也做过网站的课设任务，但那时仅仅只用了html和java语言完成了一个只有跳转功能的网站。而这次设计结合了很多不同的开发技术。比如有python、django、mysql等。

虽然我之前从未接触过python语言，但是学过之后发现python的语法确实比较简单而且方便。比起C语言，它的实现会显得比较干净简短。在开发过程中遇到的比较麻烦的事就是由于我用的是社区版的pycharm，它有很多功能没有，比如看不到某些函数的原定义在哪之类的，还有就是debug比较麻烦。不过在我写代码中最麻烦的还是python的严格规范的格式问题，有时候少空了几格程序就完全不运行了！这次设计也锻炼了我写代码中要更细心的毛病。

Django是我觉得很好用的框架，它在里面已经定义了很多很好用的功能，虽然说在设计的过程中BUG层出不穷，不过最终通过问老师同学和百度搜索都完美的解决了问题。在大二学期时我们已经学过一些数据库知识，但是MYSQL还是第一次接触。其中管理数据库的Navicate工具我也觉得非常的方便快捷。

第一次实现大型项目还是实现的比较困难，不过好在有同学和老师的帮助下最终还是顺利完成了系统测试。学习程序的过程中肯定是不那么容易的，我们必须拿出吃苦耐劳的精神，勤于发现问题、思考问题、总结问题。同时程序的语言也是一直在更新换代，我们也要及时更进技术的步伐。