NewBroadcast设计文档

项目：清华大学广播站

2014年11月

## 目录

[目录 2](#_Toc405412546)

[1 引言 3](#_Toc405412547)

[2 系统设计 4](#_Toc405412548)

[2.1 总体结构示意图 4](#_Toc405412549)

[2.2 UML序列图 4](#_Toc405412550)

[3 分层设计说明 5](#_Toc405412551)

[3.1 WEB层设计说明 5](#_Toc405412552)

[3.2 控制器层设计说明 9](#_Toc405412553)

[3.3 数据库访问层设计说明 13](#_Toc405412554)

[4 数据库设计说明 15](#_Toc405412555)

[4.1 整体ER 图 15](#_Toc405412556)

[4.2 User用户表 15](#_Toc405412557)

[4.3 ProgramGroup节目组别表 16](#_Toc405412558)

[4.4 ProgramSeries节目系列表 16](#_Toc405412559)

[4.5 Program节目表 16](#_Toc405412560)

[4.6 Source源文件表 16](#_Toc405412561)

[4.7 Comment评论表 17](#_Toc405412562)

[4.8 Praise点赞表 17](#_Toc405412563)

[4.9 Favorite收藏表 17](#_Toc405412564)

[5 设计模式 18](#_Toc405412565)

[5.1 MVC 18](#_Toc405412566)

[5.2 AOP 18](#_Toc405412567)

[6 前端技术 19](#_Toc405412568)

[6.1 栅格化布局 19](#_Toc405412569)

[6.2 异步请求 19](#_Toc405412570)

[6.3 LocalStroage 19](#_Toc405412571)

## 1 引言

清华大学广播台是校内重要的媒体之一，每周一至周五中午及下午在校园主干道上通过喇叭播放各类节目，包括新闻、书评、音乐、广播剧等等，同时也在人人网、喜马拉雅电台、蜻蜓 FM 等平台线上投放节目。广播台下共有读书、时评、资讯、访谈、音乐、体育、广播剧共计 7 个组，每组有若干节目系列，同时在校庆等重要日子全台也会一起推出一些系列节目。

清华大学广播台的网络平台开发计划旨在能够将所有广播台高质量的节目与资讯以互联网为载体统合并传播给每一位浏览该网站的用户，通过对网络平台的不断建设与完善，广播台能够以更有影响力的方式将所有作品分享给每一个喜爱校园广播节目的人；对于用户而言，网络平台则能够提供给用户更新的资讯和更自由的选择，接受信息的方式。

NewBroadcast是服务于清华大学广播站的网站项目。

|  |  |
| --- | --- |
| 开发语言 | Python、HTML |
| 框架 | Django |
| 模块数量 | 14 |
| 运行平台 | 64位系统，平台无关 |

## 2 系统设计

### 2.1 总体结构示意图

数据库

对应数据库的表项

实体层

web层

获取用户操作，向下一层发出请求

HTML页面、JS

控制器

处理请求

控制器层

数据库

访问层

根据控制器的数据需要，访问数据库

数据库访问

### 2.2 UML序列图

数据库

数据库访问

控制器

HTML、JS

1.根据用户操作向控制器发出请求

6.实现用户所需要的操作

2．根据数据需要调用数据库访问层

3.操作数据库表项

4.根据要求操作了数据库表项

5.提供给控制器相应数据

## 3 分层设计说明

### 3.1 WEB层设计说明

#### 3.1.1 总体描述

此层涵盖了HTML/ Django模板渲染和JS脚本的部分。HTML完成了各个网页自身的设计，配合Django模板渲染，展现了完整的网站视图。JS脚本负责接收用户的请求以及接收响应。

Django根据urls.py完成了网页到控制器的映射，完成了前端网页与后端处理的解耦合。JS脚本在响应了用户的请求后，提交到特定的网页进行处理，urls.py将这些网页映射到对应的控制器处理，最终根据处理结果进行跳转页面或页面响应。

#### 3.1.2 功能

在需求文档的基础上，我们对前端网页进行了相应的规划设计。具体结构如下：

#### 3.1.3 输入

JS脚本的输入为用户的操作，对于系统发出的请求。urls.py的输入为来自前端网页的URL请求。

#### 3.1.4 输出

输出为系统处理完用户的操作和请求后返回到前端网页的结果，包括数据和相应动作的执行、完成。

#### 3.1.5 流程逻辑

控制器

HTML、JS

1.根据用户操作向控制器发出请求

2.实现用户所需要的操作

#### 3.1.6 接口

#### 3.1.6.1 HTML与Django模板渲染之间的接口

在本工程前端网页开发上，我们主要采用了以下方法：根据控制器传回的数据，采用Django模板语言，生成前端网页；填充基础的Django模板模块，以达到模块复用的效果。

* {% block title %} {% endblock %} 网页标题渲染；
* {% block header %} {% endblock %} 网页头信息渲染；
* {% block body %} {% endblock %} 网页主体模块渲染；
* {% block navigation %} {% endblock %} 网页导航栏模块渲染，默认为navigation.html；
* {% block footer %} {% endblock %} 网页页脚信息渲染，默认为footer/footer.html；
* {% block script %} {% endblock %} 网页Js代码渲染；
* {% if %} {% else %} {% endif %} ，判断if后条件如果为真，渲染{% if %} {% else %}之间的内容，否则渲染{% else %} {% endif %}之间的内容；
* {% for X in Y%} {% endfor %}，每次遍历，都会渲染{% for %} {% endfor %}之间的内容。

#### 3.1.6.2 网址与控制器的映射

网页前端接受到用户的操作或者请求后，提交到指定的链接执行，而链接和控制器之间的映射关系，就由Django通过urls.py维护，每一个Action对应一个控制器。下面是由urls.py维护的网址与控制器的映射关系：

| URL | 控制器 |
| --- | --- |
| '^$' | frontpage.show\_index |
| '^upload/(?P<path>.\*)$' | django.views.static.serve |
| '^admin/' | admin.site.urls |
| '^index/$' | frontpage.show\_index |
| '^index/waterflow$' | frontpage.waterflow\_data |
| '^resource/$' | resource.show |
| '^resource/sort/$' | resource.sort |
| '^resource/filter/$' | resource.filter |
| '^resource/listall/$' | resource.list\_all |
| '^resource/groupfilter/([0-9]+)$' | resource.group\_filter |
| '^resource/getarr/$' | resource.get\_arr |
| '^program/([0-9]+)$' | program.show\_program |
| '^program/play/(.\*)$' | program.play\_program |
| '^program/praise/$' | program.praise |
| '^program/unpraise/$' | program.un\_praise |
| '^program/favorite/$' | program.favorite |
| '^program/unfavorite/$' | program.un\_favorite |
| '^program/comment/add/' | program.add\_comment |
| '^program/comment/del/' | program.del\_comment |
| '^program/upload/ajaxupload/$' | program.ajax\_upload |
| '^program/upload/$' | program.show\_upload |
| '^program/modify/([0-9]+)$' | program.show\_modify |
| '^program/modify\_program/([0-9]+)/$' | program.modify\_program |
| '^program/modify/delpic/$' | program.del\_pic |
| '^program/modify/deldoc/$' | program.del\_doc |
| '^program/delete/$' | program.delete\_program |
| '^program/get\_all\_favorites$' | program.get\_all\_favorites |
| '^program/recommand/$' | program.recommand\_program |
| '^login/do/$' | login.login |
| '^login/logout/$' | login.logout |
| '^signin/judge/$' | login.exist\_judge |
| '^signin/do/$' | login.signin |
| '^space/$' | manage.show\_space |
| '^manage/favorites/$' | manage.show\_favorites |
| '^manage/favorites/table/$' | manage.show\_favorites\_table |
| '^manage/resource/$' | manage.show\_mgrres |
| '^manage/myresource/$' | manage.show\_mgrmyres |
| '^manage/allresources/$' | manage.show\_mgrallres |
| '^manage/user/$' | manage.show\_mgruser |
| '^manage/changepassword/$' | manage.change\_password |
| '^manage/changeinfo/$' | manage.change\_info |
| '^manage/changepower/$' | manage.change\_power |
| '^manage/groupseries/$' | manage.groupseries |
| '^manage/groupseries/group/$' | manage.program\_group |
| '^manage/groupseries/series/$' | manage.program\_series |

上面定义了处理用户请求和操作的函数，可以实现WEB层与底层数据库之间的连接。

#### 3.1.7 流程逻辑

控制器

HTML、JS

1.根据用户操作向控制器发出请求

2.实现用户所需要的操作

### 3.2 控制器层设计说明

控制器层包含登陆/注册模块、管理员模块、节目资源模块等。

#### 3.2.1 总体描述

控制器层用于接收由urls.py路由过来的各种请求和操作，并且调用下一层完成功能。它接受一个http request请求，从request中获取请求或操作的信息，处理完成后返回一个包含数据的json，或是操作成功与否的结果，或是一个页面跳转。

#### 3.2.2 设计

控制器与具体的操作或请求一一对应，每一个控制器都接收由urls.py路由的http request请求，大部分的控制器都与模板层有交互。

#### 3.2.3 接口说明

下面将分模块介绍控制器的接口：

#### 3.2.3.1 首页（frontpage.py）

首页模块主要处理访问网站首页的请求和首页中瀑布流的展示。接口如下表：

|  |  |  |
| --- | --- | --- |
| 控制器 | 权限要求 | 功能介绍 |
| show\_index(http request) | 无 | 处理用户访问网站首页的请求 |
| waterflow\_data(http request) | 无 | 展示首页的瀑布流，返回一个包含首页节目的json |

#### 3.2.3.2 资源（resource.py）

资源页面展示了网站所有的节目资源，并可以按照组别、系列排序。也提供关键字搜索。

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| |  |  |  | | --- | --- | --- | | 控制器 | 权限要求 | 功能介绍 | | | |
| show(http request) | 无 | 处理访问资源页面的请求，返回结果是一个包含节目系列和节目组别的json |
| sort(http request) | 无 | 处理对节目按组别或者系列排序的请求，返回结果是一个包含排序后的节目的json |
| filter(http request) | 无 | 处理筛选满足组别、系列、关键字的节目的请求，返回结果是一个包含筛选后节目的json |
| list\_all(http request) | 无 | 处理列出所有节目的请求，返回结果是一个按照权重排序的节目的json |
| group\_filter(http request, group\_id) | 无 | 处理筛选某一组别节目的请求，返回结果是一个包含筛选后节目的json |
| get\_arr(http request) | 无 | 加载节目的具体信息，返回一个包含资源页面需要的节目各项信息的json |

#### 3.2.3.3 节目（program.py）

节目模块包括展示具体节目信息，播放节目，点赞、收藏等操作，同时还为管理员提供了上传、修改节目的接口，为超级管理员提供推荐节目的结果。

|  |  |  |
| --- | --- | --- |
| 控制器 | 权限要求 | 功能介绍 |
| show\_program(http request, program\_id) | 无 | 处理节目详细信息页面。返回结果是包含节目各种信息的json |
| play\_program(http request, program\_id) | 无 | 处理节目的播放请求。返回一个节目播放的页面 |
| praise(http request) | User | 处理对节目点赞的操作 |
| un\_praise(http request) | User | 处理对节目取消点赞的操作 |
| favorate(http request) | User | 处理对节目收藏的操作 |
| un\_favorite(http request) | User | 处理对节目取消收藏的操作 |
| add\_comment(http request) | User | 处理对节目评论的操作 |
| del\_comment(http request) | Super Admin | 处理删除节目评论的操作 |
| ajax\_upload(http request) | Worker | 处理对节目上传的操作 |
| show\_upload(http request) | Worker | 处理显示节目上传页面的请求 |
| show\_modify(http request, program\_id) | Worker | 处理显示节目修改页面的请求 |
| modify\_program(http request, program\_id) | Worker | 处理对节目修改的操作 |
| del\_pic(http request) | Worker | 删除上传和修改页面的待上传图片 |
| del\_doc(http request) | Worker | 删除上传和修改页面的待上传稿件 |
| delete\_program(http request) | Worker | 处理删除节目的操作 |
| get\_all\_favorites(http request) | User | 获取当前用户所有的收藏节目 |
| recommand\_program(http request) | Admin | 处理推荐节目的操作 |

#### 3.2.3.4 登陆/注册（login.py）

登陆注册模块提供了处理用户登录、登出、注册等操作的接口。

| 控制器 | 权限要求 | 功能介绍 |
| --- | --- | --- |
| login(http request) | 无 | 处理用户登录操作 |
| logout(http request) | 无 | 处理用户登出操作 |
| exist\_judge(http request) | 无 | 判断用户名和邮箱是否存在 |
| signin(http request) | 无 | 处理用户注册操作 |

#### 3.2.3.5 管理（manage.py）

管理模块为普通用户提供了查看个人中心、查看收藏节目、查看/修改个人信息和密码的接口，为不同权限的管理员提供了查看/修改已上传节目，查看/修改注册用户，增加/修改/删除节目组别和系列的接口。

| 控制器 | 权限要求 | 功能介绍 |
| --- | --- | --- |
| show\_space(http request) | User | 处理显示用户个人中心的请求 |
| show\_favorites(http request) | User | 处理显示用户收藏的节目的请求 |
| show\_favorites\_table(http request) | User | 处理显示用户所有收藏的节目的请求 |
| show\_mgrres(http request) | Worker | 处理显示由自己上传的节目的请求 |
| show\_mgrmyres(http request) | Worker | 处理显示所有由自己上传的节目，并提供管理的请求 |
| show\_mgrallres(http request) | Admin | 处理显示所有上传的节目，并提供管理的请求 |
| show\_mgruser(http request) | Super admin | 处理展示所有的注册用户并提供管理的请求 |
| change\_power(http request) | Super admin | 提供修改用户权限的接口 |
| change\_password(http request) | User | 处理修改密码的操作 |
| change\_info(http request) | User | 处理修改个人信息的操作 |
| groupseries(http request) | Super admin | 处理显示增加/修改/删除组别或系列页面的请求 |
| program\_group(http request) | Super admin | 提供增加/修改/删除节目组别的接口 |
| program\_series(http request) | Super admin | 提供增加/修改/删除节目系列的接口 |

#### 3.2.4 输入

Http request，int等作为输入项。

#### 3.2.5 输出

包含请求的信息的json、操作成功与否的标志、错误提示信息、页面跳转。

#### 3.2.6 流程逻辑

数据库访问

控制器

HTML、JS

1.根据用户操作向控制器发出请求

4.实现用户所需要的操作

2．根据数据需要调用数据库访问层

3.提供给控制器相应数据

### 3.3 数据库访问层设计说明

#### 3.3.1 总体描述

数据库访问层的主要功能是对数据的操作，是与数据库交互的层。由于我们采用了Django的models模块，其提供了访问数据库的各种接口，所以我们的数据库访问层完全与具体的数据库解耦。这些接口提供了对于数据库的保存、筛选、删除操作，在此基础上，我们能够完成对于user、program等对象的精确获取、条件筛选等操作，而无需考虑其具体实现。

#### 3.3.2 接口介绍

|  |  |
| --- | --- |
| 接口形式 | 功能介绍 |
| model() | 构造函数，生成model的对象 |
| model.obejects.filter() | 筛选函数，筛选符合条件的对象 |
| model.obejects.get() | 获得符合条件的单个对象 |
| model.objects.getlist() | 获得符合条件的对象列表 |
| object.save() | 保存当前对象 |
| object.del() | 删除当前对象 |

其中，model是声明在models.py的类，对应数据库中的一张表；object是model的一个对象，对应数据库表中的一个表项。

#### 3.3.3 流程逻辑

UML序列图

数据库

数据库访问

控制器

1．根据数据需要调用数据库访问层

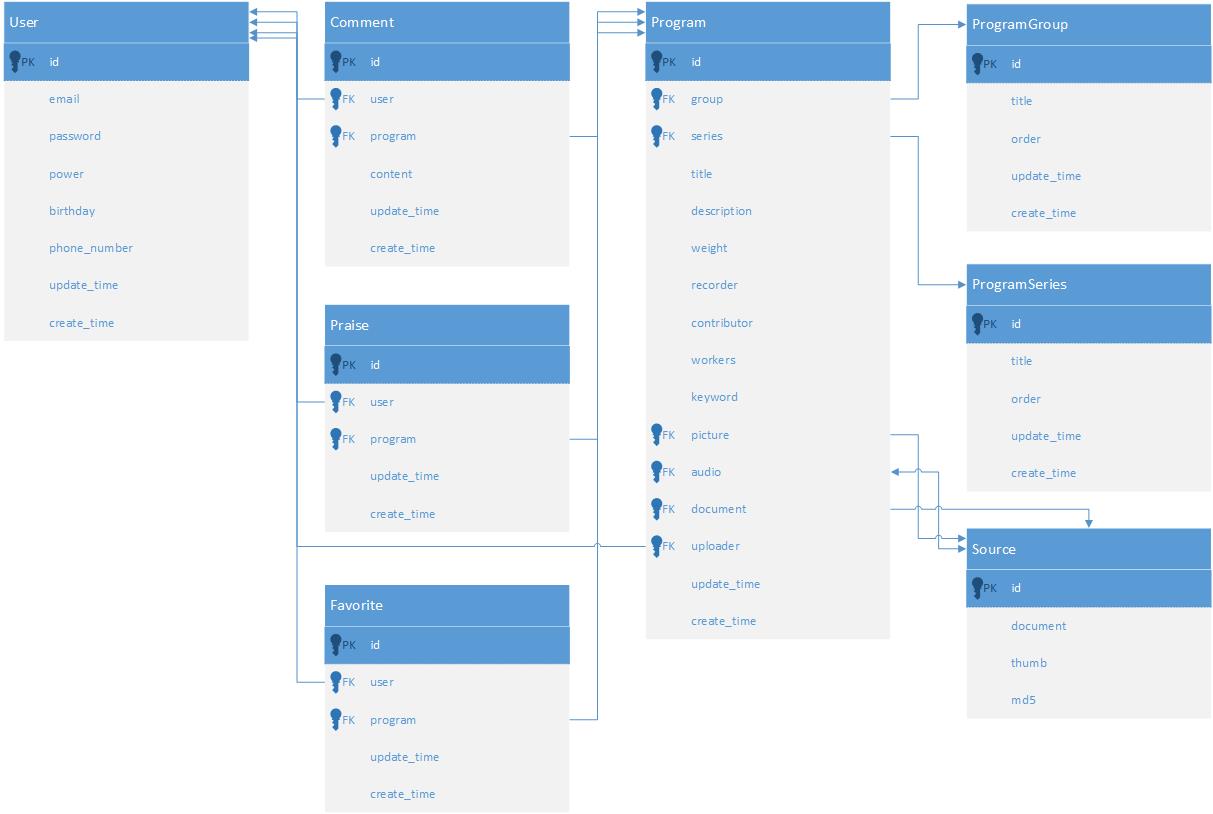
3.提供给控制器相应数据

2.操作数据库表项

4.根据要求操作了数据库表项

## 4 数据库设计说明

### 4.1 整体ER 图



我们的数据库采用关系型数据库，配置在服务器上的是MySQL。

所有的表项的主键均为id键，是DAO层将一个新表项存入数据库时自动生成的。

所有的表项均含有创建时间以及更改时间键，是DAO层将一个新表项存入数据库时自动生成的。

非空键表示存入数据时必须有值，但可以是None。必填键表示值不能为null。唯一键表示表中不能有另一个项的相同键值与之相同。

### 4.2 User用户表

记录用户的相关信息，包括邮箱、密码、昵称、权限、生日、手机等。其中邮箱、密码、权限是非空键，邮箱、昵称是唯一键。其中权限限制了用户的行为，当前SESSION中的用户的权限决定了该用户可以运行哪些服务。权限只能为用户、台员、组长、台长四种，具体见权限系统部分。

### 4.3 ProgramGroup节目组别表

记录节目组别相关信息，包括名称、在列表中的顺序等，它们都是非空键。若在列表中的顺序小于0，则在列表中不显示。

### 4.4 ProgramSeries节目系列表

记录节目序列相关信息，包括名称、在列表中的顺序等，它们都是非空键。若在列表中的顺序小于0，则在列表中不显示。

### 4.5 Program节目表

记录节目的相关信息，包括所属节目组别、所属节目系列、名称、描述、权重、播音员、节目来源、工作人员、关键字、图片、音频文件、稿件、上传者等。其中，节目组别、节目系列、上传者是外键，分别指向对应的节目组别、节目系列、用户表中的表项。图片、文稿是一个源文件表id的list，表示与之关联的图片和文稿。其中图片list中的第一个id对应的源文键表项中所存的图片会作为首页以及音乐盒中显示的图片，而所有图片都可以在节目页面查看到。音频文件是一个源文件表的id，指向与之关联的音频。关键字存储了与搜索相关的信息，即为搜索时匹配的字符串。权重即为节目推荐度，权重高的的节目会在首页中排在前面。名称、权重是非空键。

### 4.6 Source源文件表

记录源文件的相关信息，包括文件、缩略图、MD5值等。这个表中的表项可以表示图片、文稿或者音频文件。文件中包含了文件的url以及文件本身的内容。文件的存储位置由我们设置到了/upload/source中。缩略图只对于图片文件有意义，是在DAO层重载save()方法时计算并存入的，用于加速页面的加载。缩略图的位置被我们设置到了/upload/pil中。

### 4.7 Comment评论表

记录了评论的相关信息，包括发布人、相关节目、内容等，它们都是非空键以及必填键。其中发布人、相关节目是外键，分别指向了对应的用户以及节目表项，在用户以及节目之间建立了一个多对多关系。

### 4.8 Praise点赞表

记录了点赞的相关信息，包括点赞人、相关节目等，它们都是非空键以及必填键。其中点赞人、相关节目是外键，分别指向了对应的用户以及节目表项，在用户以及节目之间建立了一个多对多关系。

### 4.9 Favorite收藏表

记录了收藏的相关信息，包括收藏人、相关节目等，它们都是非空键以及必填键。其中收藏人、相关节目是外键，分别指向了对应的用户以及节目表项，在用户以及节目之间建立了一个多对多关系。

## 5 设计模式

### 5.1 MVC

我们的网站采用了MVC架构，即model-view-controller架构。Model负责封装与数据库的交互，view负责渲染用户可见的页面，而controller负责view和model之间的连接。

这种设计模式关键的优势在于各种组件都是松散结合的。这样，每个由Django驱动的Web应用都有着明确的目的，并且可独立更改而不影响到其它的部分。比如，开发者更改一个应用程序中的URL而不用影响到这个程序底层的实现。设计师可以改变HTML页面的样式而不用接触Python代码。数据库管理员可以重新命名数据表并且只需更改一个地方，无需从一大堆文件中进行查找和替换。

### 5.2 AOP

后端AOP的设计模式，即aspect oriented programming，面向切面编程。我们后台依据业务逻辑分为了数据库、DAO、服务、动作几层，使得各层之间的耦合度降低，易于分工开发和维护。

面向切面编程是通过预编译方式和运行期动态代理实现程序功能的统一维护的一种技术。AOP是OOP的延续，是函数式编程的一种衍生范型。利用AOP可以对业务逻辑的各个部分进行隔离，从而使得业务逻辑各部分之间的耦合度降低，提高程序的可重用性，同时提高了开发的效率。

## 6 前端技术

### 6.1 栅格化布局

前端采用了Bootstrap框架的栅格化布局，能够自适应不同分辨率的屏幕。

### 6.2 异步请求

前后端通讯大量使用了ajax技术，通过异步请求实现局部刷新页面等功能。

通过在后台与服务器进行少量数据交换，AJAX 可以使网页实现异步更新。这意味着可以在不重新加载整个网页的情况下，对网页的某部分进行更新。传统的网页（不使用 AJAX）如果需要更新内容，必须重载整个网页面。

Ajax 的核心是 JavaScript 对象 XmlHttpRequest。该对象在 Internet Explorer 5 中首次引入，它是一种支持异步请求的技术。XmlHttpRequest使用 JavaScript 向服务器提出请求并处理响应，而不阻塞用户。

### 6.3 LocalStroage

音乐盒使用了html5的LocalStorage本地存储进行跨页面通讯。这使已打开音乐盒的用户可以在不跳转页面的情况下刷新音乐盒的内容，增加网站的易用性，改善用户体验。