微信公众号管理平台

需求与设计文档

PRJ1 / Team imSoNervousAh

黄豪硕 胡泽聪 韦毅龙 杨天龙

目　录

1 项目基本情况

1.1 基本情况

本项目为微信公众号管理平台，甲方为清华大学。在软件工程课程中隶属于 PRJ1。

本项目可以为学校提供管理校园内学生组织公众号的网页端平台，学生用户可以用校园网账号登录本平台进行公众号的备案申请，管理员用户则可以审批申请、查看各公众号的信息并进行数据统计与分析。此外，管理员和学生用户还可以通过“站内信”进行交流沟通。

1.2 开发环境

本项目使用的框架如下：

类别	名称
前端界面与 JavaScript	Bootstrap 3.3.5 jQuery 2.1.4
HTTP 服务器	
后端网络框架	Python 2.7.10 + Django 1.8.4
后端单元测试	

1.3 团队成员与分工

本项目的开发小组为 imSoNervousAh。组员共 4 人，分工如下：

组员	分工	电子邮箱
黄豪硕（组长）	• 前后端接口设计与对接； • 单元测试。	hhs14@mails.tsinghua.edu.cn
胡泽聪	• 前端界面设计与实现； • 交互设计与用户体验优化。	huzecong@gmail.com
韦毅龙	• 整体架构设计； • 后端实现； • 数据库接口设计与实现。	wyl8899k@gmail.com
杨天龙	• 前端界面实现； • 服务器端部署。	yangtianlong111@gmail.com

1.4 浏览器兼容性

本项目的网站完美兼容 Firefox、Safari、Opera、所有 Chrome 内核的浏览器[1]，以及 IE 9.0 以上的版本。同时，网站对于 IE 8.0 版本部分兼容。网站还提供了适配移动端浏览器的

[1] 这包括了几乎所有的国产浏览器，除非当浏览器开启了 IE 兼容模式时。

版本，可以兼容各主流手机系统的较新版本。

1.5 部署说明

为了运行代码，您需要：

配置服务器

我们假定您的布署环境是 Ubuntu 14.0 以上版本的操作系统。

下载源码

您可以从 http://stash.secoder.net/projects/AN/repos/anervouswebsite/browse 上下载本项目的源代码。

安装 Python 及相关包

首先安装 Python 2.7，然后使用 `pip` 安装 `django`、`nose` 和 `coverage` 框架。命令如下：

```
sudo pip install django nose coverage
```

接着进入源码所在目录，执行如下命令：

```
cd nervous/
python manage.py runserver 8000:0:0:0 &
```

这样就能在本机运行 Web Server 了

如果直到这一步都没有问题，使用 git 切换到最近的一个 master 分支，访问 http://localhost:8000 便应该能看到登录界面

安装 nginx、配置，及服务器部署

接下来需要完成 Release 版本的搭建和 nginx 的配置，命令如下：

```
git checkout server                 #使用 git 切换到 server 分支
sudo apt-get install nginx          #安装 nginx
sudo netstat -ntlp                  #检查 nginx 是否启动
sudo /etc/init.d/nginx start        #如果没有启动，则手动启动
cp nginx.conf /etc/nginx/sites-enabled/nervous
    #将源代码目录下的 nginx.conf 复制到 nginx 配置目录下
sudo service nginx restart          #重启 nginx
python /nervous/api/bing_domain.py
    #绑定服务器域名到 nervous.wicp.net
```

访问 http://nervous.wicp.net/便能看到登陆界面。

2 架构设计

2.1 总览

网站采用 Django 构建，利用 Django ORM 操作 SQLite3 数据库。我们使用新媒体指数（gsdata）获取公众号信息。

`database/backend.py` 将数据库和抓取公众号信息的细节对网站其他部分隐藏，而提供一系列的接口供网站其他部分调用。以下将直接称这一模块为“后端”。

`api/views.py` 是网站前后端的接口。当前端提交表单或点击按钮时，会产生到 URL `/api/*`的 POST 请求，这些地址对应到了 `api/views.py` 中的 Django 渲染函数，这些函数同样通过前述后端来访问、修改数据库。

`nervous/settings.py` 是网站的总设置，以下将直接称这一模块为“总设置”。

`wechat/templates` 存放了网站的所有网页，`wechat/static` 则存放了网站的所有静态文件，包括图片、JS 文件和 CSS 样式表。

2.2 数据库

这部分主要在 `database/*`下。主要的代码文件如下：

- `models.py` 包含了各个 Django 数据模型的定义，详见“Django 数据模型”；
- `backend.py` 提供网站其他部分访问数据库的接口，详见“后端部分”；
- `daemon.py` 是网站部署完成用于定时更新的脚本。

2.3 信息抓取

这部分主要在 `api/getdata.py` 和 `api/update.py` 中。

我们使用 gsdata 提供的 API 获取公众号的各项信息。gsdata 提供的调用被我们用 Python 重写以便从后端进行调用。使用的 gsdata 账户可以在 `api/getdata.py` 当中修改。

抓取的信息包括公众号总体的信息和以天为单位的每日公众号统计信息。这些信息在数据库中的存储参见“`AccountRecord`”。

2.4 用户类别

我们将网站的使用者分为三类：**学生用户**、**管理员用户**和**超级管理员**。

学生用户

学生用户可以进行的操作有：

- 使用学生证号（或其他身份标识，由学校提供的登录接口确定）登录系统；
- 添加、撤回和删除申请；
- 对于已通过审批的公众号，可以使用站内信与管理员沟通。

学生用户在正式使用网站之前会被要求填写自己的真实姓名、电话与邮件，而这些信息会在学生提交申请时作为负责人的各项信息。没有填写信息的学生用户无法使用其他功能。

管理员用户

管理员用户可以进行的操作有：

- 审批公众号申请；
- 查看及删除已有公众号；
- 查看公众号统计信息与图表；
- 查看及发送站内信；
- 添加预警规则；
- 查看预警规则触发记录。

超级管理员

超级管理员可以进行的操作有：

- 添加及删除管理员；
- 查看网站整体统计信息；
- 修改管理员公告；
- 手动进行数据更新。

超级管理员的用户名和密码在 `nervous/settings.py` 当中由 `SUPERUSER_USERNAME` 和 `SUPERUSER_PASSWORD` 指定，其中密码为 MD5 形式。

2.5 公众号的备案申请

公众号经学生提交申请、管理员审批通过后进入数据库。在提交申请时，后端会向 gsdata 询问是否存在该公众号，如果不存在则无法提交[2]。在管理员审批通过一个公众号时，后台会开始对该公众号的第一次数据抓取[3]。

申请一共有四种状态：

状态	描述
尚未提交	学生用户在创建或修改申请之后选择“保存”，申请会成为“尚未提交”状态。

[2] 这一行为可以通过总设置中的 `settings.ALLOW_INVALID_WX_NAME` 控制。

[3] 这一行为可以通过总设置中的 `settings.UPDATE_UPON_APPROVING` 控制。

	此时申请只对学生用户可见，且可以修改。
待审批	学生用户在创建或修改申请之后选择“提交”，申请会进入“待审批”状态。 此时申请亦对管理员可见，管理员可以查看详情并进行审批。处于此状态的申请可以被学生用户撤回，成为“尚未提交”的申请。
已通过审批	管理员批准备案申请后，申请进入“已通过审批”状态。 此时申请对双方可见，其对应公众号被系统收录。学生用户还可以访问申请对应的公众号的站内信页面。 处于此状态的申请无法修改。当管理员删除申请对应的公众号时，申请也会被删除。
被拒绝	管理员拒绝备案申请后，申请进入“已通过审批”状态。 此时申请对双方可见，但是对应的公众号不会被收录。处于此状态的申请无法修改，也无法被删除[4]。

申请在各个状态之间的转移图如下：

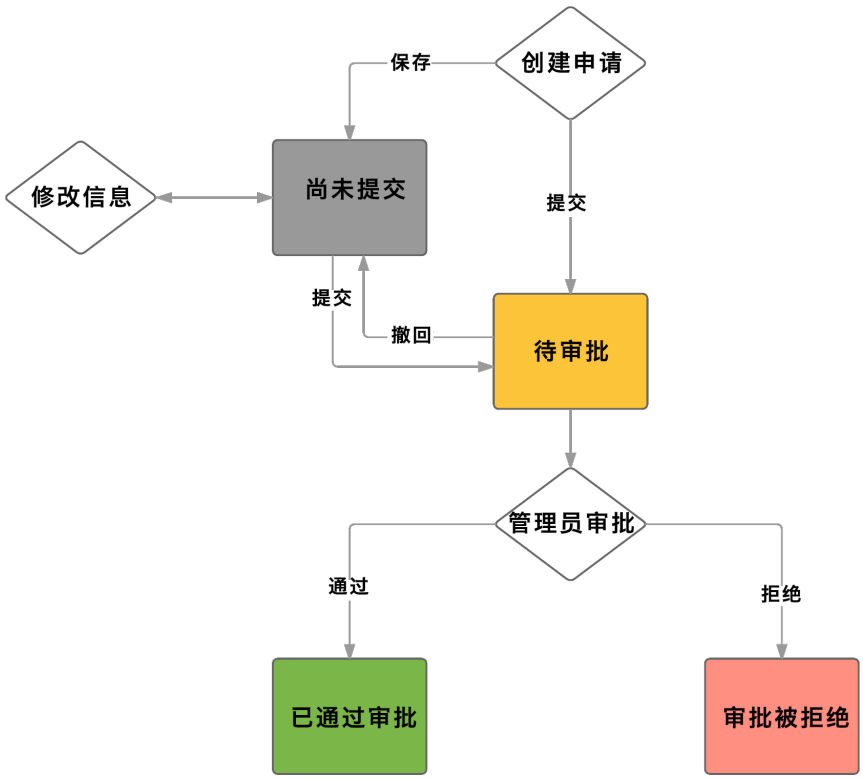


图 1 申请的状态转移图

[4] 这是目前版本在设计上的失误：由于公众号与申请一一对应，而公众号要求其微信 ID 唯一，因此，一旦申请被拒绝，学生用户无法修改申请，也无法对同一公众号重新申请，从而导致该公众号被“永久拒绝”。

3 技术细节

3.1 Django 部分

3.1.1 Django 数据模型

我们使用 Django ORM 来构建和管理数据库。定义的数据模型（model）列表如下。

Admin

表示一个管理员。管理员的定义见“管理员用户”。

域列表		
域名	描述	类型与限制
username	账户名。	字符串，长度至多为 20。
description	附加说明，通常填写其真实姓名。	字符串，长度至多为 100。
password	密码，以 MD5 形式存储。	字符串，长度至多为 32。
email	邮箱地址。	字符串，长度至多为 254。

附加说明：

- username 作为主键值，确保用户名互异；
- username 不能与超级管理员的账户名相同。

Student

表示一个学生。学生的定义见“学生用户”。

域列表		
域名	描述	类型与限制
student_id	唯一身份标识（学生证号或其他）。	字符串，长度至多为 20。
real_name	真实姓名。	字符串，长度至多为 20。
dept	系别。	字符串，长度至多为 40。
tel	电话号码。	字符串，长度至多为 20。
email	邮箱地址。	字符串，长度至多为 254。

附加说明：

- student_id 作为主键值，确保学生身份互异；
- 如果后台学校登录接口发生改变，student_id 也可以存储 Usereg 用户名；

- 电话号码和邮箱地址在存入数据库前都会经过基础的合法性验证。

`OffcialAccount`

表示一个公众号。

域列表		
域名	描述	类型与限制
`wx_id`	由字母、数字和下划线组成的公众号微信 ID。	字符串，长度至多为 50。
`name`	名称。	字符串，长度至多为 40。
`description`	简介。	字符串，长度至多为 300。
`likes_total`	总点赞数。	整数。
`views_total`	总浏览数。	整数。
`wci`	微信传播指数（WCI）[5]。	浮点数。
`update_status`	更新状态，详见附加说明。	整数。

附加说明：

- `likes_total`、`views_total` 和 `wci` 在第一次抓取数据前为空；
- `update_status` 仅被更新脚本使用，不同整数值具有如下含义：

值	描述
`0`	正常。
`1`	等待更新。
`2`	正在更新。
`3`	更新完毕，且与上次更新结果相同。
`4`	更新完毕，且与上次更新结果不同（即，有新的单日数据）。
`5`	刚刚完成一次完整的更新，等待一段时间后方能进行下一次更新。

- 如果有任何非“`0`/正常”状态的公众号，那么将无法开始新一次更新；
- 所有处于“`4`”状态的公众号在更新结束后将被运行预警规则检查。

`AccountRecord`

表示一个公众号的某一天的单日数据记录。

域列表

[5] 定义请见新媒体指数官方网站 http://www.gsdata.cn/site/usage。

域名	描述	类型与限制
account	所属的公众号。	ForeignKey。
date	记录的日期。	DateField。
likes	单日点赞数。	整数。
views	单日浏览数。	整数。
articles	单日推送文章数。	整数。
wci	当天的微信传播指数。	浮点数。

附加说明：

- 微信传播指数由新媒体指数给出的公式（详见脚注 5）直接计算得到，而这一公式可能在未来发生改变，导致和对应公众号上直接从新媒体指数抓取的最近 WCI 不同。

Application

表示一个申请。申请的处理流程参见“公众号的备案申请”。

域列表		
域名	描述	类型与限制
official_account	所申请的公众号。	OneToOneField。
user_submit	提交申请的学生账户名。	字符串，长度至多为 32。
operator_admin	通过或拒绝该申请的管理员。	字符串，长度至多为 32。
reject_reason	拒绝原因。	字符串,长度至多为 140。
status	所处状态。	字符串，长度至多为 20。
manager_name	公众号负责人的姓名。	字符串，长度至多为 30。
manager_student_id	公众号负责人的学生证号。	字符串，长度至多为 15。
manager_dept	公众号负责人的系别。	字符串，长度至多为 40。
manager_tel	公众号负责人的电话。	字符串，长度至多为 20。
manager_email	公众号负责人的邮件地址	字符串,长度至多为 254。
association	公众号所属的组织	字符串，长度至多为 30。

附加说明：

- 申请可能被多次审批，但 operator_admin 域仅会保存最后一次操作的管理员；
- status 域由于历史原因没有采用枚举类型，而包含如下的字符串之一：

值	描述
"approved"	已通过审批。

"rejected"	被拒绝。
"pending"	待审批。
"not_submitted"	尚未提交。

Article

表示一篇公众号文章。

域列表		
域名	描述	类型与限制
title	标题。	字符串，长度至多为 50。
official_account_id	所属公众号的数据库 ID。	整数。
description	描述。	字符串，长度至多为 300。
avatar_url	封面图片地址。	字符串，长度至多为 300。
url	文章地址。	字符串，长度至多为 300。
likes	点赞数。	整数。
views	浏览数。	整数。
posttime	发布时间。	DateTimeField。

Message

表示一封站内信。

域列表		
域名	描述	类型与限制
official_account	所属公众号。	ForeignKey。
category	类别，详见附加说明。	整数。
content	内容。	字符串，长度至多为 140。
processed	是否已经阅读/回复。	布尔值。
admin	发信的管理员。	ForeignKey。
time	发信时间。	DateTimeField。

附加说明：

- category 的值应当来自枚举类型 MessageCategory，其可能值为：

值	描述
MessageCategory.All	用于查询时选择所有分类。
MessageCategory.ToStudent	从管理员发往学生的站内信。

MessageCategory.ToAdmin	从学生发往管理员的站内信。

ForewarnRule

表示一个预警规则。

域列表		
域名	描述	类型与限制
account	作用范围，详见附加说明。	ForeignKey。
duration	生效时长，以天为单位。	整数。
notification	通知方式，详见附加说明。	整数。
target	预警目标，详见附加说明。	整数。
value	预警阈值，详见附加说明。	整数。
time	添加时间。	DateTimeField。

附加说明：

- account 表示预警规则生效的公众号范围，分两种情况：

值	描述
空	对所有公众号生效。
非空	对单一公众号生效。

- notification 的值应当来自枚举类型 NotificationOption，其可能值为：

值	描述
NotificationOption.Email	通过电子邮件通知。
NotificationOption.Message	通过站内信通知（尚未实现）。

- target 的值应当来自枚举类型 ForewarnTarget，其可能值为：

值	描述
ForewarnTarget.ViewsTotal	总浏览数。
ForewarnTarget.LikesTotal	总点赞数。

- target 和 value 联合起来表示预警规则内容，如 target 为 LikesTotal，value 为 30，即表示该公众号总点赞数超过 30 时触发规则。

ForewarnRecord

表示一条预警规则触发记录。

域列表		
域名	描述	类型与限制

datetime	触发时间。	DateTimeField。
account	触发的公众号。	ForeignKey。
target	预警目标。	整数。
value	预警阈值。	整数。

附加说明：

- 每当一个预警规则被某个公众号触发，就会生成一条触发记录；
- target 和 value 值含义见“ForewarnRule”。

Globals

存储一些全局唯一的信息。

域列表		
域名	描述	类型与限制
announcement	超级管理员向管理员的公告。	字符串，长度至多为 256。

附加说明：

- 采用类似单件模式的做法控制其全局唯一性。

3.1.2 Django 渲染函数

wechat/views.py 中包含了网站所有页面对应的 Django 渲染函数，其中大部分与页面和 urls.py 中的 URL 一一对应，故此处不赘述。

Ajax 请求的处理

由于站内的链接跳转绝大部分是通过 Ajax 请求完成的，因此在请求载入页面时，不需要返回整个页面，而只需要返回主页面部分即可。而在用户直接通过网址访问，或者是刷新页面的时候，又需要返回整个页面。

因此在 wechat/views.py 中定义了 render_ajax 函数，用于处理可能在 Ajax 请求中访问的链接。函数的调用方法类似 Django 自带的 render 函数，其接口如下：

```
render_ajax(request, url, params, item_id='')
```

参数列表	
参数	描述
request	Django 的 Request 对象。
url	要渲染的 HTML 文件路径。
params	Django 模板中用到的各参数。
item_id	页面对应的左侧边栏的项目 DOM ID。

函数会判断请求是否为 Ajax 请求，如果是，则渲染仅包含主页面的网页；否则渲染整个网页。因此，对于每个页面，应该存在两个文件，详见下一节“模板结构”。

而对于非 Ajax 请求，函数会在参数中补充其他必要信息。详见代码，此处不赘述。

可排序表格的渲染

关于可排序表格，请参见“可排序表格”。与其相关的 Django 渲染函数有两个：

`render_sortable``(request, items, url, params=None)`

渲染可排序表格，调用方式类似上一小节中的 `render_ajax` 函数。

参数列表	
参数	描述
`request`	Django 的 `Request` 对象。
`items`	排序、筛选前的表格项目，为 Django 的数据模型对象。
`url`	要渲染的 HTML 文件路径。
`params`	Django 模板中用到的各参数。

函数会从 GET 请求中获取由网页端传来的排序、筛选、分页要求，包括：

参数	描述
`sort_order`	排序的顺序。默认为`"desc"`（降序）。
`sort_by`	排序的键值。默认为空字符串。
`search_keyword`	搜索的关键词。默认为空字符串。
`page`	当前页码。默认为 `1`（第 1 页）。

函数还会调用下面的分页函数获得分页信息，并将信息打包进 `params` 中传入 Django 模板中，在 `common/pagination.html` 中渲染。

`get_pagination``(item_total, item_per_page, cur)`

返回分页信息。

参数列表	
参数	描述
`item_total`	表格的总项目数。
`item_per_page`	每页显示的项目数。
`cur`	当前页码。

3.1.3 Django 网页模板

本项目通过 Django 模板的继承避免了大量的代码重复。下面介绍模板的组织结构，以及 `template/`中各网页的作用。

模板结构

`base.html` 为最基础的模板，其中包含了页面的框架，包括导航栏、左侧边栏，以及主页面（详见“页面布局”）。其中包含 7 个 `block`：

block 名称	描述
`title`	页面的标题。
`subtitle`	导航栏上的副标题，即为“>”右侧的部分，详见“页面布局”中的截图。
`user-menu`	单击用户名弹出的下拉菜单。 默认只包含“登出”按钮。
`main-frame`	包含左侧边栏和主页面的整个页面。通常不更改。
`left-column`	左侧边栏的内容部分。
`main-page`	主页面。
`modals`	页面中的阻塞式窗口，属于历史残留，已弃用。关于阻塞式窗口，详见“阻塞式窗口”。

对于下述三个界面的每一个，其中都包含一个 `index.html`，继承自 `base.html`。这些模板填写了上述所有 `block`，包含有整个界面通用的副标题、左侧边栏等。

对于界面中的其他页面，均对应两个 HTML 文件[6]。比如公众号详情界面，对应 `admin/articles/articles.html` 与 `admin/articles/articles.ajax.html` 两个文件。文件名中带有“`.ajax`”的页面包含且仅包含在主页面中显示的所有内容，另一个文件则继承自对应界面的 `index.html` 文件，并用 Django 模板的`{% include ... %}`语法导入了“`.ajax`”的页面。

学生用户界面

学生用户界面对应 `student` 文件夹的中 HTML 文件，其基础模板为 `student/index.html`。其他文件如下：

路径[7]	描述

[6] 请参见“Ajax 请求的处理”。

[7] 表格中的路径均省去界面文件夹的前缀与`.html` 后缀。带星号（`*`）的路径表示该页面存在带有“`.ajax`”的版本的文件。下同。

* show_applications	首页。学生用户的所有申请的列表页面。
* modify_applications	添加/修改申请的表单页面。
info	学生用户信息填写页面。

管理员用户界面

管理员用户界面对应 admin 文件夹的中 HTML 文件，其基础模板为 admin/index.html。由于管理员用户界面文件较多，因此分文件夹列出：

admin 根目录文件	
路径	描述
* dashboard	首页。各项信息概览页面。
application_modal	申请详情的阻塞式窗口，详见“阻塞式窗口”。
applications/	文件夹。申请列表页面。
articles/	文件夹。公众号文章列表页面。
detail/	文件夹。公众号详情页面。
forewarn/	文件夹。预警规则与预警记录页面。
official_accounts/	文件夹。公众号列表页面。
statistics/	文件夹。统计信息页面。
applications 文件夹文件	
路径[8]	描述
*! applications	申请列表页面。
articles 文件夹文件	
路径	描述
*! articles	公众号文章列表页面。
detail 文件夹文件	
路径	描述
* detail	公众号详情页面的框架。
! detail_articles	公众号详情页面的文章列表板块。
detail_statistics	公众号详情页面的统计与分析板块。
forewarn 文件夹文件	
路径	描述

[8] 带感叹号（!）的路径表示该页面包含可排序表格，即存在带有“_list”和“_content”版本的文件，详见“可排序表格”。下同。

*! forewarn_records	预警记录页面。
*! forewarn_rules	预计规则页面。
official_accounts 文件夹文件	
路径	描述
*! official_accounts	公众号列表页面。
statistics 文件夹文件	
路径	描述
* statistics	统计信息页面的框架。
! official_accounts	统计信息页面的公众号列表板块。

超级管理员界面

超级管理员界面对应 superuser 文件夹的中 HTML 文件，其基础模板为 superuser/index.html。其他文件如下：

路径	描述
* admins	首页。管理员列表页面。
* manage_database	数据库信息概览页面。
* modify_announcement	管理员公告修改页面。
* update_database	数据库手动更新页面。
add_admin_modal	添加管理员的阻塞式窗口，详见“阻塞式窗口”。

其他页面

根目录文件	
路径	描述
login	网站首页。登录页面。
notfound	链接未找到时的 403 页面。
common/	文件夹。可排序表格相关页面，详见“可排序表格”。
message/	文件夹。站内信页面。
message 文件夹文件	
路径	描述
* message	站内信页面。

3.1.4 前后端接口

`api/*`用于接收前端提交表单或点击按钮时产生的 POST 请求，即充当前后端之间的接口。所有渲染函数原则上返回一个格式固定的 JSON（请见后文），用以告知前端操作是否成功。

表单的提交与返回

对于提交表单产生的 POST 请求，渲染函数会在 JSON 中指出所有没有通过合法性验证的域名称及具体原因（如果有的话）。这主要是通过下面的函数装饰器做到的：

`json_response_general_exception_decorator``(func)`

按被装饰的函数的是否抛出异常和返回值分三种情况：

函数状态	返回值

抛出异常 `e`	返回 JSON：
	`{ "status": "error",`
	`  "error_message": e.__unicode__() }`

无返回值	返回 JSON： `{ "status": "ok" }`
有返回值	以其返回值作为最终返回值。

`json_response_validation_error_decorator``(func)`

按被装饰的函数的是否抛出异常和返回值分三种情况：

函数状态	返回值

抛出 `ValidationError`	返回 JSON：
	`{ "status": "error",`
	`  "submit_method": method,`
	`  { "field_name_1": "error_message_1",`
	`"field_name_2": "error_message_2", ... } }`
	其中：
	• `method` 为前端 POST 请求当中 `method` 域的值，为`"save"`或者`"submit"`；
	• `"field_name_n"` 和 `"error_message_n"` 分别为 `ValidationError` 给出的所有合法性检测失败的域的名称和具体错误原因。

无返回值	返回 JSON：
	`{ "status": "ok",`
	`  "method": method }`

	其中 `method` 含义同上。
有返回值	以其返回值作为最终返回值。

关于网页端对于表单的处理，请参见“表单提交处理函数”。

3.2 HTML 与 CSS 部分

3.2.1 概况

在界面设计方面，本项目使用了 Bootstrap 的框架，并在此基础上使用了 Script Eden[9] 的主题 CSS。字体选择上，网页的英文字体使用了 Montserrat 和 Source Sans Pro，中文字体则使用了苹方、思源黑体、冬青黑体、微软雅黑和华文细黑[10]。

本项目的登录界面与主界面的风格是不同的。登录界面借鉴了谷歌的登录界面的设计。为了加速访问，登录界面没有使用 Bootstrap 的框架。

3.2.2 页面布局

主界面

本项目的主界面如下：

图 2 主界面的各部分

页面最顶端为导航栏（`#top-nav`），左侧为左侧边栏（`#left-column`），右侧为主页面

[9] http://scripteden.com/

[10] 并非同时使用了这些字体，而是会根据用户所安装的字体优先选择考前的字体作为网站的主要中文字体。

（`#main-page`）。导航栏始终固定不动；左侧边栏内嵌入了另外一个 DOM 容器，其内容可以单独滚动，而整个左侧边栏在页面滚动到底端后会随着主页面一起滚动，否则会一直保持固定不动[11]；主页面则拥有正常的滚动模式。

当用户点击左侧边栏中对应的项目，或者是在主页面中点击了站内链接后，新的页面会被载入到主页面上。页面载入的逻辑详见“Ajax 载入响应函数”。

阻塞式窗口

有时为了让用户优先响应一些事件，网站会弹出阻塞式窗口，如下图：

图 3 阻塞式窗口

按照 Bootstrap 中的命名规则，这一窗口的 CSS class 被称作`.modal`，而其后的暗色背景被称作`.modal-backdrop`。

[11] 请参见 `jquery/jquery.fixer.js` 中的布局逻辑。

网站中用到的阻塞式窗口有两种，一种是上图中的这种包含复杂 HTML 页面的窗口，这种窗口中的内容是存储在单独的 HTML 文件中，在需要弹出时通过载入对应 URL 加载的。另一种则是简单的对话框，如下图：

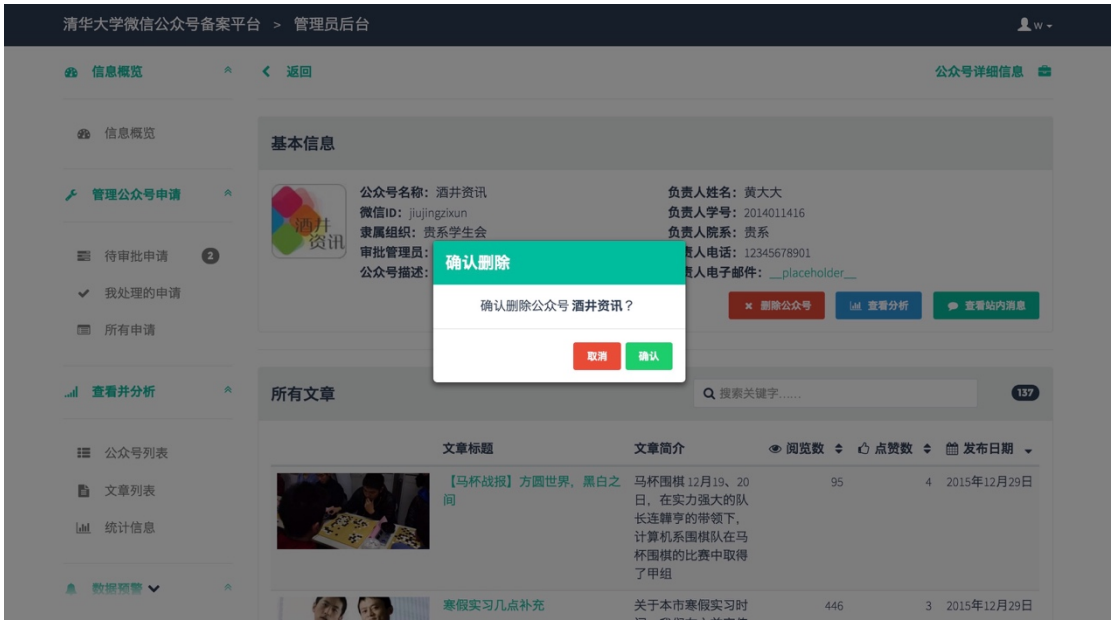


图 4 阻塞式对话框

这种对话框则只需指定标题、内容等参数即可。详见“弹出窗口函数”。

移动端布局

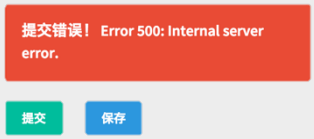
本网站也专门对移动设备做了访问优化。从移动端访问网站的效果如下：

图 5 移动端布局

移动端布局中，左侧边栏平时是隐藏的，可以点击左上角的按钮弹出。移动端布局基于 Bootstrap 的响应式布局，使用 CSS3 的 Media Query 实现，当页面宽度小于 768px 时，则会自动切换到移动端布局。

3.2.3 CSS class

本项目使用的 CSS class 大体和 Bootstrap 中规定的相同，但也设定了一些自定的规则。下面将列举各个主要的 CSS class 及其作用：

class 名称	描述	图例
`.ajax-link`	使用 Ajax 请求进行跳转的链接。 附加 DOM 属性：`data-url`，代表转到的链接。	无
`.article-avatar`	公众号详情页面中文章列表里文章的封面图像。仅截取了图片的中间一部分。	
`.badge`	指示数量的气泡，如右图所示。 附加 DOM 属性：`data-source`，代表更新数量的 URL。可选，详见“页面更新处理函数”。	
`.fake-link`	伪装成链接的元素：鼠标移上去时会变成手型指针，但是不会添加下划线。	
`.fake-link-scroll`	同上，而且点击后会将页面滚动到顶端。	
`.form-error-msg`	表单中用于提示提交错误的栏目。通常处于提交按钮与表单主体之间，在没有错误信息时会被折叠。	
`.left-column-item`	左侧边栏的菜单项目。 附加 DOM 属性：`data-url`，代表转到的链接。另见“Ajax 载入响应函数”。	

`.left-column-menu`	左侧边栏的菜单标题。	信息概览

3.2.4 表单

网页中的表单均为Bootstrap自带的表单，并使用了Validator.js做表单的网页端校验。

关于后端对于表单的处理，请参见前后端接口的“表单的提交与返回”部分；关于网页端对于表单的处理，请参见 JavaScript 的“表单提交处理函数”部分。

3.2.5 可排序表格

本项目中有多个地方需要用到可以按不同列排序的、可搜索的表格。出于软件设计中 DRY 原则的考虑，项目设计了可以复用的可排序表格元件（`.sortable`），下称“可排序表格”。其界面请见下图：

图 6 可排序表格的界面元素

可排序表格将表格分为了头部（`thead`）和内部部分（`sortable_content`），并利用 Django 模板继承来复用重复代码。其表格的核心部分包含下面的文件：

路径	描述
`common/pagination.html`	分页控件。
`common/sortable.html`	表格的核心部分。
`common/sortable_content.html`	表格的内容部分。

在复用时，需要创建两个文件，分别继承自上表中的后两个模板。以项目中的 `admin/articles` 页面为例，创建的两个文件为 `admin/articles/articles_list.html`

和 admin/articles/articles_content.html。

核心部分

admin/articles/articles_list.html 即为此处表格的核心部分。核心部分共有 4 个 block：

block 列表	
block 名称	描述
title	表格的标题。
defaults	表格的属性设置，详见下表。
head	表格的头部。只需对于每列插入一个 th 的 DOM 元素。
content-after	表格结束后插入的一段，通常为表单底部的按钮或者是表单相关的 JavaScript 等。用例请参见 admin/forewarn/forewarn_rules_list.html。
defaults 参数列表	
参数	描述
url	获取表格内容的 URL。必填。
sort_by	默认的排序关键字。
sort_order	默认的排序顺序。
search_field	搜索的关键字。如果不需要搜索，则可以省去。

defaults 的 block 实际上是从 JavaScript 代码中 JSON 常量中提出来的一块，因此所有参数均为字符串，应该按照如下格式填写：

```
<参数名>: "参数值",
```

而且最后一行结尾不得有逗号。

对于需要排序的列，需要在其头部的 th DOM 元素中添加.th-sortable 的 class，并添加 data-key 的属性。此处的 data-key 即表格每行的 Django 数据模型中，在该列显示的数据的域名。

内容部分

admin/articles/articles_content.html 即为此处表格的内容部分。内容部分一共有 2 个 block：

block 列表	
block 名称	描述
no-data-info	当表格为空时显示的内容，默认为"无数据"。
content	表格的内容。

此处“表格的内容”即为正常使用表格时会填写的内容，即一个 `tr` 的 DOM 元素代表一行。由于使用了 Django 模板渲染，通常会使用其`{% for item in items %}`的语法。

整体页面

以上两部分准备就绪后，只需在页面内需要插入表格的位置使用 Django 模板的`{% include ... %}`语法将核心部分（`admin/articles/articles_list.html`）导入，同时，将内容部分（`admin/articles/articles_content.html`）绑定到 URL，并使用 Django 渲染函数渲染即可。详情请见“可排序表格的渲染”。

3.3 JavaScript 部分

3.3.1 概况

本项目使用的两个核心框架是 jQuery 和 History.js。前者是当下最流行的库之一，它对 ECMAScript 原生操作进行了包装，提供了操作 DOM 元素的便捷方法。后者则可以借助 HTML5 的函数[12]修改浏览器的地址栏与前进/后退历史。

由于本项目中的页面切换都是依赖于 Ajax 实现，避免了实际的链接跳转，因此浏览器是不会修改地址栏与前进/后退历史。这样一来，地址栏里的 URL 会一直保持为第一个页面的 URL，此时如果用户刷新页面，则会回到第一个页面，与期待的响应不符。前进、后退也类似。借助于 History.js 库，本项目在使用 Ajax 请求完成页面切换的同时，也能正确响应刷新、前进、后退的操作。

本项目使用的其他 JavaScript 框架有 FusionCharts 图表套件、Validator.js，以及一些 GitHub 上的开源库。在此向各位库作者表示感谢。

另外，在下文中各文件的目录中，可能还存在一些同名，但是以`.min.js`结尾的文件。这些文件是对应的 JS 文件压缩之后得到的小体积 JS 文件。在实际部署使用时，应当使用这些 JS 文件。

3.3.2 文件列表

路径	描述
`base/js/charts.js`	渲染 FusionCharts 图表。详见“FusionCharts 图表”部分。
`base/js/referrer-killer.js`	突破防盗链限制下载图片。在公众号详情界面显示文章图片时使用。
`base/js/scripts.js`	核心部分。详见下文“核心部分”。
`bootstrap/js/*`	Bootstrap 相关的文件。

[12] 如果浏览器不支持 HTML5，则会通过在 URL 后加 hashtag（`#`）的方法实现类似功能。

fusioncharts/* fusioncharts/themes/*	FusionCharts 图表相关的文件。详见下文“FusionCharts 图表”部分。
ie/*	兼容 IE 的相关文件。其中包括旧版本的 jQuery，以及实现 CSS3 Media Query 的库。
jquery/jquery.js	jQuery 的核心文件。
jquery/jquery.fixer.js	基于 jQuery 的简易布局插件。在确定左侧边栏的位置时使用。
jquery/jquery.history.js	修改浏览器的地址栏与前进/返回记录。由于 Ajax 请求不会更改页面 URL，因此需要使用这一插件手动修改。否则，在用户尝试后退、前进，甚至是刷新的时候，将无法触发预期的正确操作。
jquery/jquery.scrollto.js	基于 jQuery 的滚动插件。在需要滚动到指定位置时使用。
md5/md5.js	MD5 加密。在需要发送密码时使用。
validator/validator.js	表单校验插件。

3.3.3 核心部分

Ajax 载入响应函数

loadContent(url, params, item_selector, load_params, callback)

通过 Ajax 请求实现链接跳转，并将内容载入到#main-page 上。

参数列表		
参数[13]	类型	描述
* url	string	需要载入的目标 URL。
params	JSON	Ajax 请求的附加信息。
item_selector	string	URL 对应的左侧边栏中的项目。如果不存在则为 undefined。
load_params	JSON	载入时的选项。
callback	function()	函数执行完毕时调用的回调函数。
load_params 参数列表		
参数	类型	描述

[13] 带星号（*）的参数为必填参数，即不能为 undefined。下同。

replace	boolean	是否是对页面的更新，而非载入新的页面。
anim	boolean	是否使用淡入淡出的动画效果

loadContentOn(container, url, params, load_params, callback)

通过 Ajax 请求实现链接跳转，并将内容载入到指定的 DOM 对象上。

参数列表		
参数	类型	描述
* container	string	待载入内容的 DOM 对象的选择器。
* url	string	需要载入的目标 URL。
params	JSON	Ajax 请求的附加信息。
load_params	JSON	载入时的选项。
callback	function()	函数执行完毕时调用的回调函数。
load_params 参数列表		
参数	类型	描述
anim	boolean	是否使用淡入淡出的动画效果

loadContentOfItem(item, load_params, callback)

通过 Ajax 请求载入左侧边栏中项目对应的页面。

参数列表		
参数	类型	描述
* item	string	需要载入的左侧边栏的选择器。
load_params	JSON	载入时的选项。
callback	function()	函数执行完毕时调用的回调函数。
load_params 参数列表		
参数	类型	描述
replace	boolean	是否是对页面的更新，而非载入新的页面。
anim	boolean	是否使用淡入淡出的动画效果

displayContent(data, params, container, callback)

将 Ajax 请求得到的页面内容载入到 DOM 中。

参数列表		
参数	类型	描述

* data	string or object	载入的内容。可以是 HTML 字符串，或者是 DOM 对象。
params	JSON	载入时的选项。
container	string	待载入内容的 DOM 对象的选择器。 默认[14]为"#main-page"。
callback	function()	函数执行完毕时调用的回调函数。

params 参数列表		
参数	类型	描述
anim	boolean	是否使用淡入淡出的动画效果。
scroll	boolean	是否将页面滚动到待载入对象的顶部。

表单提交处理函数

handleFormPost(form_selector, post_url, params)

通过 Ajax 请求实现表单的提交与处理。详见前后端接口的“表单的提交与返回”与 HTML 界面的“表单”部分。

参数列表		
参数	类型	描述
* form_selector	string	需要处理的表单的选择器。
* post_url	string	表单提交的发送 POST 请求的 URL。
params	JSON	表单处理的选项。

params 参数列表		
参数	类型	描述
success_callback	function(data)	表单提交成功（POST 请求成功）时调用的回调函数。
error_callback	function(data)	表单提交失败（POST 请求失败）时调用的回调函数。
success_message	function(data)	表单提交成功时，用于显示提示信息的函数。函数应当返回一个字符串，即为提示的信息。如果返回的字符串为空，则会使用默认的函数生成提示信息。如果需要显示空字符

[14] 当传入参数的值为 undefined 时，则使用默认参数。下同。

		串，则需要返回"#no_message#"。 默认函数的行为如下： • 表单校验成功时不显示信息； • 表单校验失败时，若只有单条错误信息，则显示错误信息； • 表单校验失败时，若有多条错误信息，则不显示（"#no_message#"）； • 其他情况，显示"提交出错，请再次检查您填写的信息。"。
before_submit	function(event)	在提交表单之前调用的回调函数，通常用于处理表单内容。

页面更新处理函数

initAjaxPage(container)

为 DOM 对象中需要使用 Ajax 请求完成跳转的链接（.ajax-link）进行单击事件的绑定，通常在刷新页面之后需要对刷新的 DOM 对象调用。

参数列表		
参数	类型	描述
* container	string	包含需要绑定的链接的 DOM 对象（通常为刚刚刷新的 DOM 对象）的选择器。

loadComplete()

刷新页面内的数字气泡（.badge），通常在刷新页面之后调用。

resizeComponents()

页面宽度发生变化时的处理函数。

函数会调整页面的最小高度（min-height），并触发左侧边栏的滚动事件，以更新其坐标。同时，函数还会调整对话框的暗色背景（.modal-backdrop）的高度，因此在弹出对话框的函数中也调用了此函数。

弹出窗口函数

showConfirmModal(title, message, one_button, callback)

弹出 Bootstrap 风格的对话框（而非浏览器的 alert 对话框）。

可以弹出只包含一个按钮的信息对话框，也可以弹出包含两个按钮的确认对话框。对于后者，单击确认按钮时会调用指定的回调函数。

参数列表		
参数	类型	描述
* title	string	对话框标题。
* message	string	对话框内容。
one_button	boolean	是否弹出只包含一个按钮的信息对话框。默认为否。
callback	function(event)	单击“确定”按钮时调用的回调函数。实际实现中，即为绑定到“确定”按钮的事件处理函数。

showModal(url, id)

从指定的 URL 中载入并弹出 Bootstrap 风格的阻塞式窗口。

参数列表		
参数	类型	描述
* url	string	阻塞式窗口的 URL。
* id	string	弹出的窗口的 DOM 对象 ID。

图表绘制函数

drawCharts(selected_charts)

载入 DOM 对象中包含的 FusionCharts 图表。详见“FusionCharts 图表”部分。

参数列表		
参数	类型	描述
* selected_charts	jQuery object	包含 FusionCharts 图表的 jQuery 对象。默认为$(".tab-pane object")。

初始化函数

initLeftColumn()

绑定左侧边栏相关的事件。具体处理的事件包括：

- 绑定移动端界面的弹出左侧边栏按钮的点击事件；

- 处理项目标题右端的箭头的方向的切换；
- 绑定左侧边栏内部的滚动事件；
- 为侧边栏中非选中项目绑定点击事件；
- 使用 jQuery.fixer 库完成左侧边栏的页面内位置处理。

辅助函数

`callRepeated``(callback, cycles, time)`

重复调用若干次给定的函数。

参数列表		
参数	类型	描述
* callback	function()	需要调用的函数。
* cycles	number	调用的次数。
* time	number	两次调用的时间间隔。

`animate``(item, animation)`

对 DOM 对象施加单次的 Animate.css 动画（默认情况下 Animate.css 动画会循环）。详见 Animate.css 官方网站 https://daneden.github.io/animate.css/。

参数列表		
参数	类型	描述
* item	jQuery object	需要施加动画的的 jQuery 对象。
* animation	string	Animate.css 动画的名称。

`removePx``(str)`

从字符串中删去末尾的"px"，并转换成数字返回。常用于处理 CSS 的 height 一类的属性。

参数列表		
参数	类型	描述
* str	string	需要处理的字符串。如果字符串本身即为数字，则不作处理。

3.3.4 FusionCharts 图表

与 FusionCharts 相关的 JavaScript 文件有 9 个：

路径	描述
`base/js/charts.js`	渲染 FusionCharts 图表。
`fusioncharts/startrender.js`	指示开始渲染的文件。详见下文。
`fusioncharts/fusioncharts*`（共 3 个） `fusioncharts/themes/*`（共 4 个）	FusionCharts 图表相关的文件。

项目中用到了 FusionCharts 的页面只有一个：公众号的分析界面（`admin/detail/detail_statistics.html`）。这一页面中亦有一段 JavaScript，其主要作用是初始化页面内的图表组件，并载入其它几个 JS 文件。

由于 FusionCharts 的 JS 文件体积庞大，需要相当长的载入时间，因此需要使用非阻塞的载入方式。这段 JavaScript 中使用的方法是将 JS 文件的 URL 动态写入文档头部，从而实现非阻塞顺序载入。

为了在载入所有 JS 文件后再开始绘制图表，还需要最后载入一个 JS，其中包含开始绘制的指令。这也就是 `fusioncharts/startrender.js` 的用途。

而 `base/js/charts.js` 中也只有一个函数：

`renderChart``(container, type, data, params, callback)`

在制定的 DOM 对象绘制 FusionCharts 图表。

参数列表		
参数	类型	描述
`* container`	`jQuery object`	待绘制的 DOM 对象。
`* type`	`string`	图表的类型。详见节末的网址。
`* data`	`JSON`	图表的信息。亦详见节末的网址。
`params`	`JSON`	图表的其他附加信息，如长宽等。亦详见节末的网址。
`callback`	`Function(jQuery)`	完成绘制后调用的回调函数。

图表的信息则是在 Django 模板渲染时就处理完成，存入页面的。具体来说，信息在 Django 渲染函数中处理为 JSON 格式，并转换为字符串存入待绘制 DOM 对象的 `data-json` 属性中。图表的类型则存在 `data-type` 属性中。上一节中的 `drawCharts` 函数会读取 DOM 对象的这两个属性，并调用 `renderChart` 函数完成绘制。

关于图表信息的格式，请参考公众号详情页面的 Django 渲染函数 `admin_show_official_account_statistics`，或者访问 FusionCharts 官网 http://www.fusioncharts.com/dev/chart-attributes.html。

3.4 后端部分

Applications（申请部分）

`get_applications`()

返回所有申请。

`get_applications_by_status`(`status`)

返回所有状态为 `status` 的申请。

`get_pending_applications`()

返回所有待审批的申请。

`get_applications_by_user`(`username`)

返回所有由用户 `username` 提交的申请。

`get_applications_by_admin`(`username`)

返回所有由管理员 `username` 审批的申请。

`get_application_by_id`(`id`)

返回数据库 ID 为 `id` 的申请。

`add_application`(`dic`)

用词典 `dic` 里的信息建立申请和公众号并存入数据库。

`dic` 可以包含如下键值：

键值	描述
`name`	公众号名称。
`description`	公众号简介。
`wx_id`	公众号微信 ID。
`manager_name`	公众号负责人姓名。

manager_student_id	公众号负责人学生证号。
manager_dept	公众号负责人系别。
manager_tel	公众号负责人电话。
manager_email	公众号负责人邮件地址。
user_submit	提交申请的学生账户名。
association	公众号所属的组织。
status	申请的状态。
reject_reason	拒绝申请的原因。

student_modify_application(dic)

用词典 dic 里的信息修改已有的申请，用于学生用户编辑申请内容。

除了 add_application 中 dic 的键值外，dic 还应该包含键值 id，代表要修改的申请。

modify_application(app)

用词典 app 里的信息修改已有的申请，用于管理员审批申请。

dic 应包含如下键值:

键值	描述
id	要修改的申请的 ID。
status	申请的状态。
operator_admin	审批申请的管理员名称。

del_application(id)

删除数据库 ID 为 id 的申请。

recall_application(id)

撤回数据库 ID 为 id 的申请。

OfficialAccount（公众号部分）

get_official_accounts()

返回所有审批通过的公众号。

get_official_account_by_id(id)

返回数据库 ID 为 id 的公众号。

get_official_accounts_with_unprocessed_messages(category)

返回具有 category 类别的未读站内信的公众号。类别的定义参见“Message”。

del_official_accont(id)

删除数据库 ID 为 id 的公众号。

Admin（管理员账号部分）

add_admin(username, md5_password, email, description)

添加一个用户名为 username，密码的 MD5 为 md5_password，电子邮件地址为 email，附加说明为 description 的管理员。

del_admin(username)

删除用户名为 username 的管理员，如果其存在。

get_admins()

返回所有管理员。

get_admin_emails()

返回一个包含所有管理员的邮件地址的列表。

check_admin(username, password)

询问数据库中是否有(username, password)的管理员用户—密码对。返回一个布尔值。

Student（学生账号部分）

get_student_by_id(student_id)

返回账户名为 student_id 的学生。

set_student_information(student_id, dic)

将词典 dic 内的内容写入账户名为 student_id 的学生当中。

dic 应包含如下键值：

键值	描述
real_name	真实姓名。
dept	系别。
tel	电话。
email	电子邮件地址。

Article（公众号文章部分）

get_articles(sortby=SortBy.Time,
order=SortOrder.Descending,
start_from=0,
count=10,
filter=None)

将所有文章按照 sortby 指定的键值排序，排序顺序为 order，经 filter 筛选后，返回一个元组(cnt, articles)，其中：

参数	描述
cnt	筛选后文章的总数。
articles	筛选后文章当中从 start_from 开始的 count 个。

sortby 的值应来自于 database/models.py 当中定义的 SortBy 枚举类型，其可能值如下：

值	描述
SortBy.Time	按照发表时间排序。
SortBy.Likes	按照点赞数排序。
SortBy.Views	按照浏览数排序。

order 的值应来自于 database/models.py 当中定义的 SortOrder 枚举类型，其可能值如下：

值	描述
SortOrder.Ascending	按升序排序。
SortOrder.Descending	按降序排序。

filter 是一个词典，可以含有以下键值：

键值	描述
official_account_id	限定文章所属的公众号的数据库 ID。
article_title_keyword	限定文章标题的关键字。
posttime_begin	限定文章发表日期起点。
posttime_end	限定文章发表日期终点。

其中 posttime_*应为 DateTime 类型。

get_articles_by_official_account_id(id)

返回所有属于数据库 ID 为 id 的公众号的文章。

Message（站内信部分）

get_messages(category=None,
official_account_id=None,
only_unprocessed=None)

返回按照条件筛选后的站内信的列表：

参数	描述
category	限定类别，定义见“Message”。
official_account_id	限定所属公众号的数据库 ID。
only_unprocessed	若为真则限定未读的站内信。

process_all_messages(official_account_id, category)

将所有属于数据库 ID 为 official_account_id 的且类别为 category 的站内信标为已读。category 定义同上。

add_message(category, official_account_id, content, admin_name=None)

添加一个具有以下属性的站内信：

参数	描述
category	类别，定义同上。
official_account_id	属于的公众号的数据库 ID。
content	站内信内容。
admin_name	如果是从管理员发往学生的，应当包含发信管理员名称。

AccountRecord（公众号数据记录部分）

get_records(official_account_id, day_start, day_end)

返回数据库 ID 为 official_account_id 的公众号在 day_start 和 day_end 间的每日统计信息。其中后两者应为 DateTime 类型。

get_views(account, day_start, day_end)

意义同上，但返回对应记录列表的浏览数列表。

get_latest_record(official_account_id)

返回数据库 ID 为 official_account_id 的公众号最晚的一次每日记录。如果不存在将抛出异常。

ForewarnRule（预警规则部分）

add_forewarn_rule(dic)

添加一个具有词典 dic 中声明的属性的预警规则。

dic 应当包含以下键值：duration、notification、target 和 value。各值的意义参见“ForewarnRule”。

除此之外，如果 dic 中包含非空的 account_name 值，那么预警规则仅对具有该名称的公众号生效，否则对全体公众号生效。

modify_forewarn_rule(dic)

类似前一个函数，但 dic 应额外包含一个域 id 指定要修改的预警规则的数据库 ID。

get_forewarn_rules()

返回所有预警规则。

get_forewarn_rule_by_id(id)

返回数据库 ID 为 id 的预警规则。

del_forewarn_rule(id)

删除数据库 ID 为 id 的预警规则。

get_forewarn_records()

返回所有预警规则触发记录。

delete_expired_rules()

返回所有已经超过预定生效时长的预警规则。

ForewarnRecord（预警记录部分）

forewarn_record_from_rule(rule, account)

产生一条“预警规则 rule 被公众号 account” 触发的预警记录，存入数据库并返回。

report_forewarn_record(record)

用预警规则指定的通知方式将预警记录 record 通知给管理员[15]。

check_all_forewarn_rules()

检查所有预警规则。

update_progress()

返回一个 0～100 之间的数表示当前进行的更新的进度。

updating_account_name()

返回正在更新的公众号的名称。

update_all()

更新所有公众号的信息，并对抓取到新信息的公众号检查预警规则。

Globals（全局信息模块）

get_globals()

[15] 目前只实现了通过邮件的方式通知。

返回全局唯一的 Globals 对象（如果没有则会创建一个）。

modify_announcement(announcement)

修改管理员公告为 announcement。

get_announcement()

返回当前管理员公告。

4 总结

通过本次项目，我们收获了很多。

一开始选择这个“微信公众号管理系统”的项目，是由于我们组的兴趣所致——我们组一致的想法是，写出一个能让我们自己满意的网站，可喜的是，最后我们做到了。

我们组的队名叫做 imSoNervousAh，其实是由四个组员的名字首字母组成的。原意是“我好慌呀”。这也表达了我们刚刚选这门课时的状态，作为一门大三上的“硬课”，在初上大二的我们看来，似乎还有很多东西不会，和多学习了一年的学长们的差距还是挺大的，亟需恶补知识。

刚刚选到这个项目的时候，我们便花了大量的时间进行前期的准备，最后我们选择采用我们都熟悉的 Django 来写网站。在熟悉了 Stash 和 Jira 的用法之后，我们一开始的框架搭建相当地顺利，没有什么磕磕绊绊地就完成了。实际上，我们用很快的时间就完成了基本的功能的实现，这个是我们万万没有想到的。

但是在进一步地考虑界面，以及更加人性化的方面时，我们却遇到了很多的麻烦。在前端上，我们需要现场熟悉一些工具的用法，经常为了一个小小的细节而大改一晚上。在后端上，我们对网上现成的模板进行了整合重构，包括重新实现了从 gsdata 抓取数据的 API 等等。这些都是对我们团队合作能力的巨大考验。

在构建整个项目的过程中，我们学会了从用户的角度去分析问题，也更加注意了代码的规范性和可读性。作为一个团队，我们每周都有固定的时间来交流实现思路和难点，大家集思广益，提出了自己的看法和观点。在和助教（甲方）交流的过程中，我们也初步地了解到了实际软件开发流程中可能遇到的形形色色的问题，学会了许多实用的工具的使用，收获了一份宝贵的情谊。

这些收获，想必会是我们大学生活中弥足珍贵的一段回忆。