开发指南

《软件工程》MOOC 微信抢票开发实战

目录

1 快速上手 2

1.1 环境准备 2

1.1.1 数据库（MySQL） 2

1.1.2 Python 2

1.2 申请微信公众平台测试号 2

1.3 版本库与配置文件 2

1.4 运行与调试 3

1.5 微信接入 4

2 数据库设计 5

2.1 用户表 5

2.2 活动表 5

2.3 电子票表 6

2.4 潜在优化 6

3 基础封装 6

3.1 微信接口 6

3.1.1 View 6

3.1.1.1 WeChatLib 7

3.1.1.2 CustomWeChatView 7

3.1.2 Handler 7

3.1.2.1 WeChatHandler 8

3.2 前端接口 9

3.2.1 APIView 9

3.3 静态文件 9

4 补充 10

4.1 创建管理员账号 10

4.2 获取与同步抢票菜单 10

5 部署 10

5.1 环境准备 10

5.2 后端程序 10

5.3 静态文件 10

6 测试 10

6.1 功能测试 10

6.2 性能测试 11

1 快速上手

1.1 环境准备

本实战项目，可以在 Windows、Linux、OS X 等操作系统环境下进行开发，不同操作系统的环境准备可能存在差异，请注意甄别。

1.1.1 数据库（MySQL）

本项目选用 MySQL 作为后端数据库。你可以自由选择是使用本机 MySQL 还是连接到开发服务器的 MySQL。如果使用本机 MySQL，就需要在本机进行 MySQL-Server 的配置；如果在开发服务器上使用 MySQL，首先开发小组需要有一台公用的开发服务器，然后在开发服务器上安装 MySQL-Server。

1.1.2 Python

本项目的源码为 Python 3 的源码，你需要安装 Python 3。除了基本的系统包外，还需要安装 django 和 mysqlclient 两个第三方 Python 包。可以用 pip 工具方便地安装这两个包，运行 pip install –r requirements.txt 命令即可自动安装 requirements.txt 文件内指定的所有 Python 包及其适合的版本。

1.2 申请微信公众平台测试号

在 http://mp.weixin.qq.com/debug/cgi-bin/sandbox?t=sandbox/login 使用自己的微信号扫码登录即可使用微信公众平台测试号，测试号开放了微信公众平台的所有接口，只是限制了每个账号至多 100 人关注，对于我们的开发和测试以及足够了。

1.3 版本库与配置文件

每个小组需要从原始版本库（https://github.com/ThssSE/WeChatTicket）fork 出子版本库，作为全组的开发版本库。开发前，需要先将子版本库 clone 至本地。

本工程在工作目录需要有一个名为 configs.json 的配置文件，开发中可以放置在项目根目录（即与 manage.py 同级目录）。configs.json 里都是一些与环境相关或与安全相关的配置项，为 JSON 格式的一个 object，每个字段对应一个配置项。有以下配置项：

字段	内容	举例
SECRET_KEY	安全配置，Django 的 SECRET_KEY[1]	长度为 50 的随机串
DEBUG	是否为开发环境	开发环境为 True，生产环境为 False
IGNORE_WECHAT_SIGNATURE	是否忽略微信的 Signature 检验	大多数情况下为 True，在性能测试时可以临时设置为 False

[1] 关于 Django 的 SECRET_KEY，详见 https://docs.djangoproject.com/en/1.9/ref/settings/#std:setting-SECRET_KEY

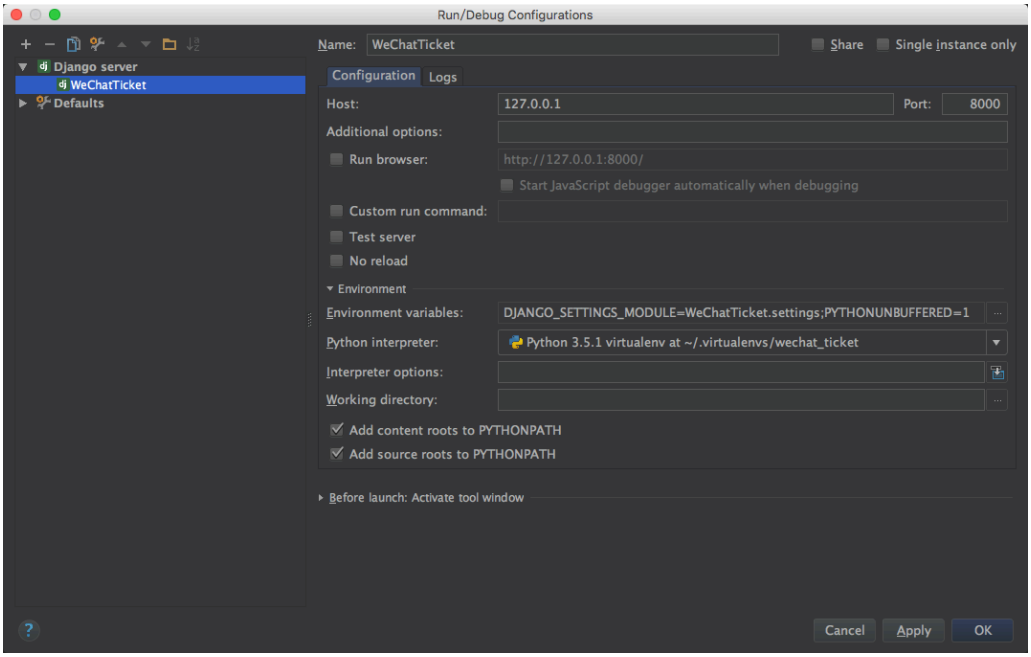
WECHAT_TOKEN	微信公众平台的配置项 Token	随机串
WECHAT_APPID	微信公众平台的信息 appID	在测试号登录后可以看到
WECHAT_SECRET	微信公众平台的信息 appsecret	登录测试号可查
DB_NAME	MySQL 数据库名称，即 database name	
DB_USER	数据库登录用户名	开发环境可以直接用 root，生产环境一般为特定数据库用户
DB_PASS	数据库登录的密码	
DB_HOST	数据库连接地址	
DB_PORT	数据库连接端口	3306
SITE_DOMAIN	后端服务器的域名，包含协议名	http://some.domain.com

一个 configs.json 对应一个环境（开发环境、测试环境、生产环境等），且该文件已经加入.gitignore。需要特别注意的是，涉及到安全问题，千万不要把这个文件提交到 git 库。

1.4 运行与调试

为了方便进行开发与调试，推荐使用 PyCharm 进行工程管理。通过自己的学生信息在 https://www.jetbrains.com/student/可以申请注册成为学生开发者，从而免费使用集成了 Django 支持的 PyCharm Professional。

使用 PyCharm 打开你的 git 版本库，它能自动辨认为 Django 项目并自动创建运行和调试配置，如下图所示（如果没有，请自行配置）。



1.5 微信接入

微信接入前，需要熟悉微信服务器与我们的开发服务器或生产服务器的通信方式以及具体消息接口，文档详见 http://mp.weixin.qq.com/wiki/home/index.html。

我们可以用微信接入我们的开发环境，从而方便开发调试。首先需要确定你当前的网络环境是否拥有公网 IP。用百度搜索“ip”可以看到当前本机接入互联网使用的公网 IP，如下图。

接着，在 PyCharm 的运行和调试配置中修改 Host 为 0.0.0.0 接受来自任何来源的请求，然后通过 http://your.ip.address:8000 尝试访问，如果访问得到 Django 的 404 页面，说明你当前的网络环境是直接使用独立 IP 接入互联网的。

如果你的当前网络环境给你的本机分配了独立 IP，可以将你的 Django 运行配置修改为 0.0.0.0:80，运行后将 http://your.ip.address/wechat 填入微信测试号的接口配置信息的 URL 项，并将 configs.json 的 WECHAT_TOKEN 配置项填入 Token 项，即可完成接入。值得注意的是，在 Linux 或 OS X 环境下，要监听 80 端口需要 root 权限，将会带来额外的安全风险，建议通过 nginx 等方式进行请求转发，Django 程序仍运行于 8000 端口。

如果你的当前网络环境并未给你的本机分配独立 IP，则一般需要通过代理来实现开发环境的微信接入。可以使用拥有独立 IP 的外网机器进行反向代理，也可以使用花生壳等代理工具进行内网映射（例如 http://www.mamicode.com/info-detail-1307548.html，http://wiki.jikexueyuan.com/project/java-wechat/5.html 等）。

在部署环境（生产环境）下，服务器一定拥有独立 IP，接入方式与开发环境无异。

2 数据库设计

本项目的数据库模型定义，全部在 wechat.models 模块中。通过 python manage.py migrate 可以进行数据表的自动创建或修改。如果你对本项目原先设计的数据库进行了任何改动，都需要先通过 python manage.py makemigrations 创建相应的数据自动迁移脚本，然后再通过 python manage.py migrate 真正地进行相关数据操作。具体这些操作的作用和原因，在 Django 的官方文档中有详细的说明：
https://docs.djangoproject.com/en/1.9/topics/migrations/

2.1 用户表

本项目主要有两类用户：后台管理员、微信用户。对于后台管理员，我们使用 Django 自带的用户管理功能（https://docs.djangoproject.com/es/1.9/topics/auth/），所有的 Django User 均视为后台管理员，可以进行后台操作。对于每个微信用户，微信服务器会提供他们的 OpenID，在同一个微信公众平台上，每个用户都有独特的 OpenID。（需要注意的是，在不同的微信公众平台上，即使是同一个用户也可能拥有不同的 OpenID，不过本项目只涉及单一的微信公众号，因此这一点对本项目没有影响。）

用户表为 wechat.models.User，具体字段及其含义与限制如下表所示。

字段	含义	限制
open_id	OpenID	64 位以内的字符串，唯一性，索引字段
student_id	该用户已绑定的学号，未绑定时为空串	32 位以内的字符串，唯一性，索引字段

2.2 活动表

活动表存储活动的详情信息，定义为 wechat.models.Activity，具体字段及其含义与限制如下表所示。

字段	含义	限制
name	活动名称	128 位以内
key	活动代称	64 位以内，索引字段
description	活动介绍	
start_time	时间类型，活动开始时间	
end_time	时间类型，活动结束时间	
place	活动地点	256 位以内
book_start	时间类型，抢票开始时间	索引字段
book_end	时间类型，抢票结束时间	索引字段
total_tickets	整数类型，总票数	
status	整数类型，活动当前状态	取值为 STATUS_DELETED、STATUS_SAVED、STATUS_PUBLISHED，分别代表已删除、已保存未发布、已

		发布。
pic_url	活动配图的 URL	256 位以内
remain_tickets	整数类型，活动剩余票数	

2.3 电子票表

电子票表存储电子票的信息，每一条数据代表一张电子票。定义为 wechat.models.Ticket，具体字段及其含义与限制如下表所示。

字段	含义	限制
student_id	拥有者的学号	32 位以内，索引字段
unique_id	电子票的唯一 ID	64 位以内，索引字段，唯一性
activity	外键，Activity	
status	状态	取值为 STATUS_CANCELLED、STATUS_VALID、STATUS_USED，分别代表已退票、有效票、已使用

2.4 潜在优化

上述的数据库设计存在一些影响性能的因素，若开发进展顺利，可考虑进行一些优化从而提高系统性能。可考虑的潜在优化包括且不限于：

1. **去除外键**

   外键会导致在数据表的查询操作时将多张表合并查询，影响性能。可以考虑将 Ticket 表的 activity 外键约束去除，对应的在数据操作时要自行保证正确性。

2. **去除 Ticket 表的索引**

   在抢票或退票时，会频繁对 Ticket 表进行操作，在查询时，该表的索引可以大大加快针对 student_id 和 unique_id 的查询效率，但在插入时，每条插入的数据也都需要建索引，这会降低插入性能。请结合抢退票的实际数据操作，在保证正确性的前提下，权衡决定 Ticket 表的索引字段。

3 基础封装

在本项目中，已经针对微信接口、前端接口、静态文件等进行了一些基础封装，二次开发可以在这些封装的基础上进行。

3.1 微信接口

微信接口的封装模式为 View-Handlers，对于 URL 为 /wechat，将由 wechat.views.CustomWeChatView 接收处理，在其中会依次调用其 handlers 成员中定义的各个 wechat.wrapper.WeChatHandler 子类对象的 check 方法，若 check 方法返回 True，认为该请求被该 WeChatHandler 接受，进而调用其 handle 方法（且不再调用其他 WeChatHandler）。

3.1.1 View

该 View 类继承自 wechat.wrapper.WeChatView，在父类中已经实现了包括检查

signature（http://mp.weixin.qq.com/wiki/8/f9a0b8382e0b77d87b3bcc1ce6fbc104.html）、解析 XML 数据、调用 handlers 逻辑等，具体实现无需关心。

3.1.1.1 WeChatLib

WeChatView.lib 为 wechat.wrapper.WeChatLib 实例对象，可能会用到的方法如下。

get_wechat_access_token() -> access_token
raise WeChatError

获取微信公众平台的 access_token，**请注意，该方法会将 access_token 在内存中缓存，若上次获取的 access_token 未过期，将直接返回而不重新获取**。关于 access_token，详见 http://mp.weixin.qq.com/wiki/14/9f9c82c1af308e3b14ba9b973f99a8ba.html。

get_wechat_menu() -> button
raise WeChatError

本项目不使用微信公众平台的个性化菜单功能，只是用朴素的自定义菜单。因此该方法直接返回无个性化菜单情况下的 menu.button 字段内容（如微信公众平台文档所述，是一个 list），详见 http://mp.weixin.qq.com/wiki/5/f287d1a5b78a35a8884326312ac3e4ed.html。

set_wechat_menu(data) -> None
raise WeChatError

设置公众号的自定义菜单，data 为官方文档所述格式的数据，详见 http://mp.weixin.qq.com/wiki/10/0234e39a2025342c17a7d23595c6b40a.html

3.1.1.2 CustomWeChatView

在 CustomWeChatView 中对与菜单管理相关的功能进行了一些封装，可能会用到的方法如下。

update_menu(activities=None) -> None
raise WeChatError

更新菜单，根据 activities 设置“抢票”栏的各个活动子菜单。activities 为 Activity 对象的 list 或 QuerySet。若 activities 为 None，则自动读取 Activity 表未结束抢票的活动，按抢票结束时间从小到大排序，至多 5 个活动。

此外，CustomWeChatView.event_keys 还定义了不同的菜单项点击的事件 key（菜单项的点击事件也是会发送给微信公众平台后端服务器的，这在官方文档中有说明，详见 http://mp.weixin.qq.com/wiki/7/9f89d962eba4c5924ed95b513ba69d9b.html）

3.1.2 Handler

前文已经说了，本项目的微信接口封装模式为 View-Handlers，会按顺序依次尝试调用各个 handlers 的 check 方法，第一个返回 True 的 handler 视为接受请求，进而调用其 handle 方法，完成整个微信消息的处理。

本项目中主要通过继承 WeChatHandler 的子类来定义各个 handler。具体可研究

wechat.handlers 模块，在其中已经有帮助、解绑、绑定、无抢票活动等多个示例 WeChatHandler。

3.1.2.1 WeChatHandler

WeChatHandler 的一些成员如下说明。

- logger：通过 Python 自带的 logging 模块得到的 Logger，用于输出日志，详见 logging 模块的说明。
- input：解析后的微信消息，dict 对象，包含 FromUserName、ToUserName、CreateTime、MsgType、MsgId 等字段，根据消息类型不同还会有其他特定字段，见 http://mp.weixin.qq.com/wiki/17/f298879f8fb29ab98b2f2971d42552fd.html
- user：发送该消息的微信用户对应的 User 对象，若该用户未绑定，则其 student_id 将会是空串。
- view：调用方的 CustomWeChatView 对象。

下面对 WeChatHandler 的一些可能会用到的方法做一些说明。

check() -> bool

check 方法，检查微信消息是否可以由当前 Handler 处理，若接受，返回 True 即可。在 WeChatHandler 的子类中必须实现该方法。

handle() -> str

handle 方法，根据微信消息进行相应处理，返回的 str 为直接返回给微信服务器的 XML 格式的内容。

is_text(*texts) -> bool

检查消息是否为特定内容的文本消息。例如“帮助”、“help”等。

is_event_click(*event_keys) -> bool

检查消息是否为特定的菜单点击。例如“BOOKING_EMPTY”、“SERVICE_HELP”等。

is_event(*events) -> bool

检查消息是否为特定的事件。例如“subscribe”、“unsubscribe”等。

is_text_command(*commands) -> bool

检查消息是否为形如“COMMAND QUERY”格式的文本消息。例如“抢票 活动 1”、“退票 活动 2”等。

reply_text(content) -> str

回复特定的文本消息，只需提供内容即可得到相应的 XML 文本，可直接作为 handle 方法的返回值。

reply_news(articles) -> str

回复图文消息，articles 为 dict 对象的 list，支持 Title、Description、PicUrl、Url 等字段参数，详见 http://mp.weixin.qq.com/wiki/1/6239b44c206cab9145b1d52c67e6c551.html#.E5.9B.9E.E5.A4.8D.E5.9B.BE.E6.96.87.E6.B6.88.E6.81.AF

可得到 XML 文本，直接作为 handle 方法的返回值。

reply_single_news(artivle) -> str

单条图文消息，article 为 dict 对象，如 reply_news 所述，可得到 XML 文本，直接作为 handle 方法的返回值。

get_message(name, **data) -> str

从 templates 的 messages 文件夹中读取相应的模板文件（html 格式），并渲染得到 str。可参考 messages 文件夹中的已有模板文件，以及在实例 handlers 中的使用。

url_*() -> str

以 url_开始的各个方法，是前端页面的反向路由，如果在你回复给用户的消息中需要用到新的链接，建议给 WeChatHandler 增加新的 url_方法并调用，不要直接将 URL 写死在字符串或模板中。

3.2 前端接口

前端接口基类为 codex.baseview.APIView。

3.2.1 APIView

APIView 的实例对象有一个 input 成员，是 dict 类型，为自动解析得到的 Request 的参数，例如通过 self.input['name']即可得到 name 参数。如果是上传文件，例如获取 image 字段对应的文件，只需访问 self.input['image']即可。

此外，APIView 有 check_input 方法可用于检查请求的参数是否充分。而 get 和 post 方法，则对应处理 GET 和 POST 请求。关于前后端接口的更多开发细节，请参见《前后端接口》文档。

check_input(*keys) -> None
raise InputError

建议在 get 或 post 方法一开始就调用。keys 为该接口调用必须提供的字段。

3.3 静态文件

本项目使用的所有前端文件均为静态文件，存储于项目的 static 文件夹中。在开发环境下，本项目会将这些静态文件托管于 WeChatView.views.StaticFileView。在生产环境下（即 configs.DEBUG==False），为了获得更好的性能，该 View 将弃用，请将静态文件托管于 nginx 等服务器程序。

4 补充

4.1 创建管理员账号

管理员账号即 Django User，通过 python manage.py createsuperuser 即可依照引导完成创建。

4.2 获取与同步抢票菜单

抢票菜单经常需要同步，例如应该要自动从抢票菜单中移除过期的抢票活动。通过 python manage.py syncmenu 即可自动更新抢票菜单。

如果需要在服务器上查看菜单，通过 python manage.py getmenu 即可在终端或命令提示符窗口内打印出当前菜单的详情。

5 部署

5.1 环境准备

除了 1.1 介绍的开发环境外，在生产环境下还需要通过服务器程序和 Python 的 WSGI 接口来部署 Django 项目。我们推荐使用 nginx+uWSGI 的组合。

具体可以参考 https://docs.djangoproject.com/en/1.9/howto/deployment/wsgi/uwsgi/ http://www.runoob.com/django/django-nginx-uwsgi.html 等关于如何安装 uWSGI、配置 uWSGI 运行 Django 程序、在 nginx 中配置 uWSGI 的说明。

在生产环境下，你需要一套与开发环境不同的 configs.json，对于数据库、微信私密信息等，要特别注意保护。

5.2 后端程序

在 nginx 中需要将/wechat、/api/a/*、/api/u/* 这三个 URL 配置为 uWSGI 进行管理。

5.3 静态文件

在 nginx 中，需要配置静态文件目录为项目目录下的 static 文件夹，并设置默认文件为 index.html。

6 测试

6.1 功能测试

本项目目前没有任何自动化测试。你在开发中可以适当编写一些单元测试脚本。此外，功能测试在开发的任何阶段都是必须的，通过手动或自动的方式进行功能测试，才能在一定程度上减少项目在未来使用中可能出现的问题或隐患。

6.2 性能测试

本项目的性能测试可以主要关注于抢票功能上。可以通过 JMeter 等工具来进行性能测试。若是进行抢票功能的性能测试，则测试的接口为/wechat，可以在 configs.json 中设置 IGNORE_WECHAT_SIGNATURE=True 来更方便地进行测试。