基于 ThinkCMF 的微信公众号服务系统设计与实现

专业：计算机科学与技术　　学生：林开兴　　指导老师：池灵达

摘要

随着互联网技术的发展，电商成为人们购物渠道的主流。由于电商的冲击，线下实体店的生存成为了问题。本微信公众号服务系统是作为插件在一个基于社群的泛电商系统中运行。这个泛电商系统主要是为了解决拥有固定或特定群体或社群的线下实体店，比如各类俱乐部，为他们提供一个发布活动、产品等他们需要发的东西的平台，利用他们固定或特定的群体或社群，帮助他们抵挡电商带来的冲击，使得他们在互联网发达的环境中能够生成下去。

微信是当前中国乃至世界上使用最广泛的社交平台之一，而微信公众号也成为了一个商机，许多商户利用微信公众号吸收粉丝，在微信公众号上发布最新产品的消息，并放置自己发布的产品的链接，达到盈利的目的。本课题所设计并实现的微信公众号服务系统是将微信公众号作为一个吸收粉丝、推广活动或产品和盈利的工具，使得用户在一整个泛电商系统中可以一站式发布活动或产品，将其发布的活动或产品通过微信公众号推广出去。本微信公众号服务系统还可以帮助微信公众号达到吸粉的作用，当一个没有关注该公众号的用户通过微信端想要报名活动或购买产品的时候，会先让该用户关注该公众号后再进行报名或购买，从而为该公众号留存用户。

本微信公众号服务系统主要包括“微信基本功能模块”和“微信公众号功能模块”，每个模块是一个插件。它主要是对微信公众号基本信息和功能的管理，如微信支付，微信红包，自定义菜单，素材上传，群发消息等。

关键词：微信公众号　　ThinkCMF　　微信支付　　微信红包

目 录

1 绪论

1.1 项目研究背景

电商已经成为一个主流的交易形式，由于电商带来的巨大冲击力，实体店的生存岌岌可危。但是由于有一些实体店有固定的社群，比如各类俱乐部有其自己的会员，这些会员对于俱乐部的信任度是非常高的，由于淘宝等电商平台的店铺种类已经很多，这些俱乐部不可能在这些电商平台中生存下去。所以如果有一个基于社群的泛电商系统上线，泛电商包括活动报名、订购、产品等，当俱乐部在该系统中开了属于本俱乐部的店铺，并通过微信公众号来宣传，那么比起淘宝等网上店铺，这些会员会更愿意在俱乐部的店铺上买。

微信公众号可以让商家在微信平台上实现和特定群体的文字、图片、语音、视频的全方位沟通、互动。微信公众号的功能有自动发送消息、自动回复消息、管理成员、菜单管理等。微信公众号的这些功能可以通过公众号后台进行管理，也可以通过官方提供的接口进行管理。本课题主要是设计并实现一个微信公众号服务系统，帮助使用泛电商系统的客户更好的实现盈利。

1.2 国内外研究现状

根据调查发现目前微信第三方平台有非常多，杂且乱。杂是指随便的一个三四个人的开发团队便可以开发出一个微信第三方平台，乱是指这些微信第三方平台的开发结构不好，不仅 BUG 一堆，连后期的运维都有一定的难度。

目前，客户比较多的微信第三方平台有微盟、微店、有赞等。微盟主要是针对电商、餐饮、广告、移动办公等行业开发出各种各样的应用，并接入微信公众号，使得微盟的客户在有一定应用可以使用的情况下接入微信公众号，扩大自己的宣传力度和盈利范围；微店主要是针对电商，将电商接入微信公众号，微店的客户可以在微店中开一个属于自己的店铺，然后接入到自己的微信公众号，使用微信公众号的一系统功能进行推广和盈利；有赞主要是针对大型的电商，还包括了批发，将这些功能接入到微信公众号。无论这些微信第三方平台所要实现的目的是什么，对于他们的客户来说，微信公众号都是一种宣传以及盈利的工具，所以开发微信公众号各种功能接口是非常重要的。

1.3 主要研究内容

本课题的主要研究内容是开发一个微信公众号服务系统，对微信公众号的一些常用功能进行统一管理。

这些常用功能包括微信公众号基本配置功能、微信公众号支付功能、微信公众号提现功能、微信公众号自定义菜单功能、微信公众号素材上传功能、微信公众号群发消息功能、微信公众号吸粉功能、微信公众号获取用户 openid 以及基本信息功能、微信公众号获取 accesstoken 功能、微信公众号接入服务器功能等。

由于本项目属于快速开发阶段，为了更好的保证泛电商系统的上线以及使用，本课题仅针对最常用的功能进行开发。对于自定义菜单功能仅仅只能创建跳转 URL 事件；对于素材上传功能仅仅只能上传图文素材；对于群发消息功能仅仅只能针对所有人群发。

1.4 项目研究目的以及意义

本课题是作为插件在一个泛电商系统中运行，为了更好的让客户能够在使用该系统时获得更多的推广和盈利，微信作为当前中国以及世界上使用最广泛的社交平台之一，微信公众号已经是一个吸收粉丝、帮助推广、实现盈利的必不可少的平台。本课题致力于打造一个微信公众号服务系统，将本系统作为一个泛电商系统中的插件，帮助客户更好的使用这个泛电商系统。

本课题的难点是本系统作为插件在整个架构上运行，但是本插件只是对于微信公众号的各种 API 的封装，对于其他插件想要使用这些 API，就需要调用本插件提供的接口，所以将本插件封装成接口提供给其他插件使用是本课题的难点。本课题不但让本人了解到 ThinkPHP 开发框架，还了解到 ThinkCMF 系统的结构及二次开发的方法，还让本人了解微信公众号的接入方式及 API。

2 相关技术介绍

2.1 ThinkPHP 介绍

ThinkPHP 是一个免费开源的，快速、简单的，面向对象的轻量级 PHP 开发框架，遵循 Apache2 开源协议发布，是为了简化企业应用开发和敏捷 Web 应用开发而诞生[1]。

ThinkPHP 可以支持 windows/Unix/Linux 等服务器环境，需要 PHP5.0 以上版本支持，支持 MySql、PgSQL、Sqlite 多种数据库以及 PDO 扩展，ThinkPHP 框架本身没有什么特别模块要求，具体的应用系统运行环境要求视开发所涉及的模块[2]。

作为一个整体开发解决方案，ThinkPHP 能够解决应用开发中的大多数需要，因为其自身包含了底层架构、兼容处理、基类库、数据库访问层、模板引擎、缓存机制、插件机制、角色认证、表单处理等常用的组件，并且对于跨版本、跨平台和跨数据库移植都比较方便。并且每个组件都是精心设计和完善的，应用开发过程仅仅需要关注您的业务逻辑[3]。

ThinkPHP 框架系统中使用了 MVC 架构，MVC 包括了模型(Model),视图(View)和控制(Controller)。ThinkPHP 中的 MVC 提供了一种敏捷开发的手段,使得开发者能够迅速地开发出稳定的企业级网站[4]。

2.2 ThinkCMF 介绍

ThinkCMF 是一款基于 PHP+MYSQL 开发的中文内容管理框架。ThinkCMF 提出灵活的应用机制，框架自身提供基础的管理功能，而开发者可以根据自身的需求以应用的形式进行扩展。每个应用都能独立的完成自己的任务，也可通过系统调用其他应用进行协同工作。在这种运行机制下，开发商场应用的用户无需关心开发 SNS 应用时如何工作的，但他们之间又可通过系统本身进行协调，大大的降低了开发成本和沟通成本[5]。

在 ThinkCMF 框架下，一个可由用户扩展定制的 CMS（内容管理系统）已经初具雏形—它自带完整的系统后台管理模块，内置了灵活的用户管理、界面菜单管理、文章管理、页面管理等功能[6]。

ThinkCMF 自身层次非常清晰，逻辑也相当的严谨，特别是系统自带的 protal 应用非常适合 PHP 初学者使用。采用了国内优秀的开源 php 框架 ThinkPHP 使得 ThinkCMF 具备了优秀的性能以及良好的安全性。

3 需求分析

3.1 可行性分析

3.1.1 经济可行性分析

目前电商系统的普及，严重影响了线下实体店的生存，但是目前的电商针对的是所有用户，而没有针对特定用户的电商系统，比如针对拥有固定群体或社群的实体店的电商系统。基于社群的泛电商系统针对的是拥有固定群体或社群的实体店，比如各类俱乐部，这些实体店由于线上电商的冲击，使得他们无法在线下生存，但是他们所拥有的固定群体或社群，是可以去挖掘其经济意义的市场。俱乐部一般都会有活动、产品等需要发布，但是在产品这一块，如果俱乐部能有用一套线上系统帮助他们发布和推广到他们拥有的群体或社群中，这将有利于他们的生存。这些群体或社群对与俱乐部的信任是非常高的，但是由于电商的普及、线上购物的方便、自己所在的俱乐部没有线上电商，他们更愿意在其他电商如淘宝这一类地方去购买他们需要的产品，如果这些俱乐部拥有自己的电商系统，由于信任度的问题，这些群体或社群就会更愿意在自己所在的俱乐部购买产品。所以基于社群的泛电商系统拥有极大的市场，而帮助客户推广的微信公众号服务系统也有其存在的重要性。

3.1.2 技术可行性分析

随着 Web 技术越来越发达，实现网页网站的技术也越来越多，并且越来越简单，近几年出现的 PHP、JSP、ASP.NET 等技术的使用者越来越多，各有其优点，使得微信公众号服务系统的技术可行性得到认可，开发一个微信公众号服务系统的门槛也就越来越低。目前，PHP 语言是开发 Web 技术中较为简单的语言，在国内 ThinkPHP 又是一个被广泛认可的框架系统，而 ThinkCMF 又是从 ThinkPHP 延伸出来，也得到了广泛的认可，所以选用 ThinkCMF 来作为其技术可行性。

3.2 功能需求分析

随着微信的普及，微信的用户数量越来越多，微信公众号也就被作为一种营销手段。本课题所设计实现的微信公众号服务系统旨在帮助使用基于社群的泛电商系统的客户能将他发布的所有东西，如活动、产品等，推广到微信公众号上。

本课题对于微信公众号较为普遍使用的功能进行设计与实现，如公众号接入服务器、获取 accesstoken、获取用户信息、自定义菜单、群发消息、公众号支付、现金红包、素材上传等。其中由于时间的关系，目前自定义菜单仅仅能够实现设置一二级菜单和跳转 URL 事件、群发消息仅仅能够针对所有用户群发、素材上传仅仅能够上传图文素材，但是这些功能在目前就已经够给用户使用的空间。

3.3 数据需求分析

通过对微信公众号服务系统的功能需求分析，可以了解到本系统需要实现的功能，本系统所涉及的主要数据有：微信公众号基本信息数据、微信用户自动登录数据、微信公众号支付数据、微信公众号红包数据、自定义菜单数据、群发消息数据、素材上传数据。下面分别分析这些数据需求：

（1）微信公众号基本信息数据

微信公众号的基本信息数据是本系统其他功能的基础。

（2）微信用户自动登录数据

微信用户自动登录数据主要是从微信服务器获取用户的数据后进行自动注册或登录操作，方便用户在后续的一系列操作中可以不用注册或登录。这些数据也是基于社群的泛电商系统其他功能的基础。

（3）微信公众号支付数据

微信公众号支付是基于社群的泛电商系统中最常用的功能之一，在用户提交订单后，服务器生成订单数据并请求微信服务器的支付，当用户支付成功后，将支付数据写入数据库，方便日后查询等。

（4）微信公众号红包数据

微信公众号红包主要是用于用户提现时使用，当用户提现时将微信公众号的红包数据写入数据库，方便日后查询等。

（5）自定义菜单数据

自定义菜单是微信公众号最常用的功能之一，当用户在本系统中设置好自定义菜单后，本系统将数据发送给微信服务器，微信服务器接收后更改自定义菜单，而保留自定义菜单的数据是为了方便用户修改或删除自己的自定义菜单。

（6）素材上传数据

素材上传功能是群发消息功能的基础，素材上传可以上传文字素材、图片素材、图文素材等，当用户在本系统中上传了素材后，本系统将素材数据发送给微信服务器，当请求成功后将素材数据写入数据库，方便群发消息功能使用。

（7）群发消息数据

群发消息是微信公众号最常用的功能之一，群发消息后在半小时内可以对该消息进行删除，所以当用户在本系统中群发消息时，将群发消息数据写入数据是非常有必要的。

4 系统设计

4.1 系统结构设计

本系统是基于 ThinkCMF 框架系统来设计实现的，由于 ThinkCMF 框架系统本身就已经使用到了 MVC 结构，所以在最底层的系统结构中本人并没有进行改变，而且在模块层次上对系统的结构进行了改变。

一个插件就相当于一个模块，在插件中已经有了 MVC 结构，但是在 controller 中既有 model 又有对于 view 的操作也有对于数据的检验。由于这样子对于后期的运维不方便，耦合度太高，所以在插件中增加了 Logic 文件夹，里面的每个 Logic 是一个静态类，只用来处理数据，比如数据的增加、修改、删除、查询等，将这些功能都封装成方法，在 controller 中只要引用后使用 Logic 的名称就可以直接调用其中的方法，这样就将耦合度降低了，controller 中只处理数据的检查和 view 的操作。

4.2 功能模块设计

系统主要分为两个模块设计，一个是微信基本功能模块，另一个是微信公众号功能模块。

微信基本功能模块分为前台设计和后台设计，前台主要是普通用户配置自己的微信公众号基本信息、在配置好基本信息后将自己的微信公众号接入服务器、获取 accesstoken、获取用户基本信息、微信支付和微信红包，后台主要是系统管理员配置系统默认的微信公众号基本信息。

微信公众号功能模块只有前台设计，主要是普通用户在配置成功微信公众号基本信息并接入服务器后，可以在系统中使用自定义菜单、素材上传、群发消息功能。

在对本系统进行了需求分析，以及功能模块的分析，本系统的系统功能模块图，如下图 4-1 所示：

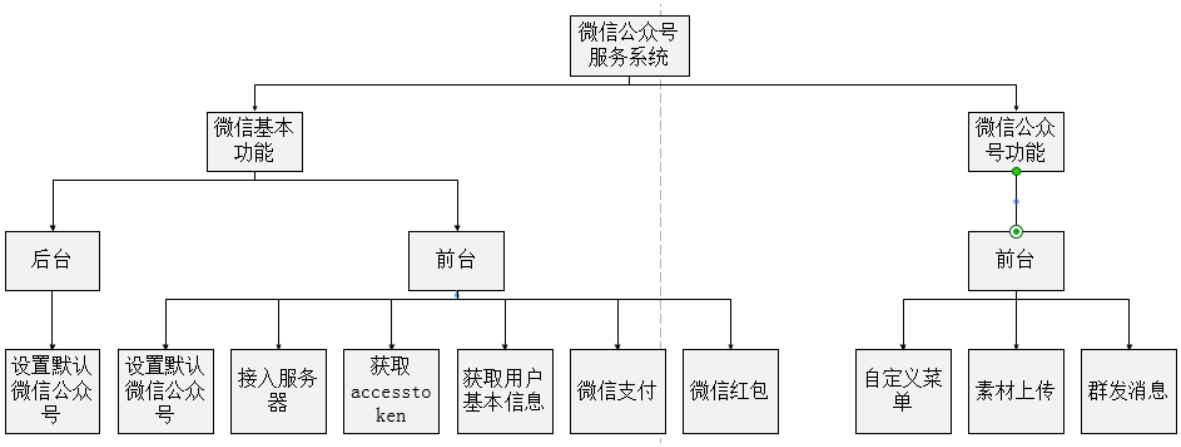

图 4-1 系统功能模块图

4.2.1 微信基本功能模块

微信基本功能模块要实现的功能包括配置微信公众号基本信息、微信公众号接入服务器、获取 accesstoken、获取用户基本信息、微信支付和微信红包。下面分别设计这些功能：

（1）配置微信公众号基本信息

这个功能分为前台和后台，前台普通用户点击“设置微信公众号”按钮后分为三个步骤让用户填写微信公众号相关的基本信息，在第二步骤还需上传微信公众号二维码，为了在点击报名后若该用户没有关注该公众号时显示该公众号的二维码，方便用户直接扫描二维码关注公众号。后台系统管理员管理微信基本功能模块时填写系统默认的微信公众号基本信息，不需要上传微信公众号二维码，为了某些微信公众号不能获取用户信息时即无法自动登录时使用系统默认的微信公众号来获取用户信息。流程图如图 4-2 所示：

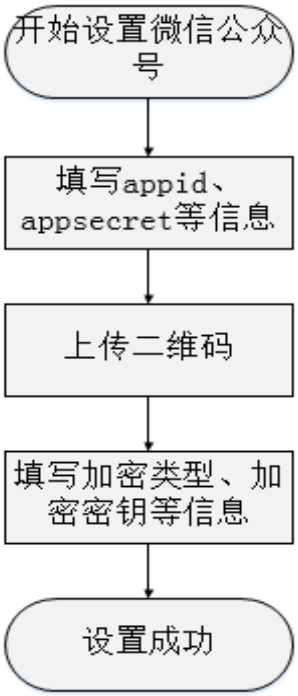

图 4-2 配置微信公众号基本信息

（2）微信公众号接入服务器

在配置微信公众号基本信息的第二步，系统会显示出用户应该在微信公众号管理后台的配置服务器中所要填写的信息，并提示用户进行填写并验证服务器的有效性。

（3）获取 accesstoken

一个微信公众号需要使用 accesstoken 时，当系统缓存中没有该微信公众号的 accesstoken 或该微信公众号的 accesstoken 过期时，向微信服务器请求并获取返回的数据，写入系统缓存中。流程图如图 4-3 所示：

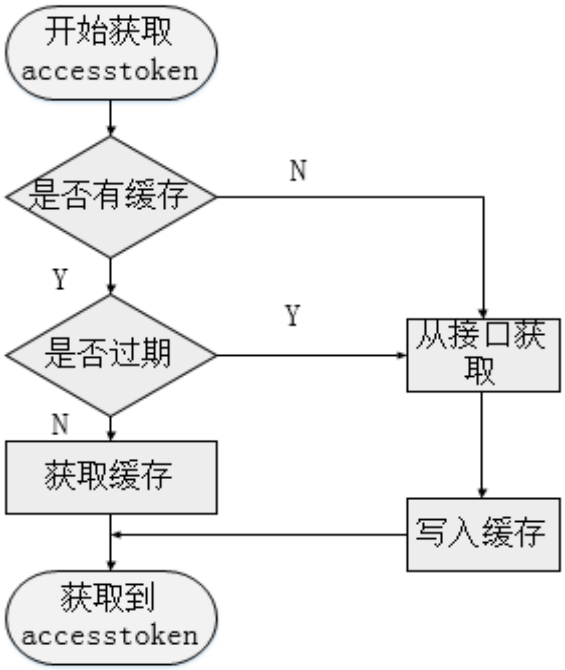

图 4-3 获取 accesstoken

（4）获取用户基本信息

当用户进入到基于社群的泛电商系统时，会先让用户授权微信公众号，授权成功后系统便能够获取到用户对应的 openid 和 accesstoken，并向微信服务器请求，获取用户基本信息。流程图如图 4-4 所示：

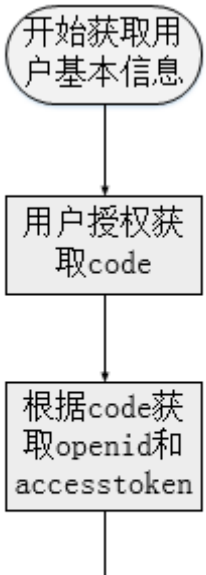

图 4-4 获取用户基本信息

（5）微信支付

当用户提交订单后，系统会自动生成订单，并请求微信服务器，获取微信支付相关数据并显示支付数据给用户，等待用户支付成功后，将最终支付成功的数据写入数据库。微信支付的时序图如图 4-5 所示：

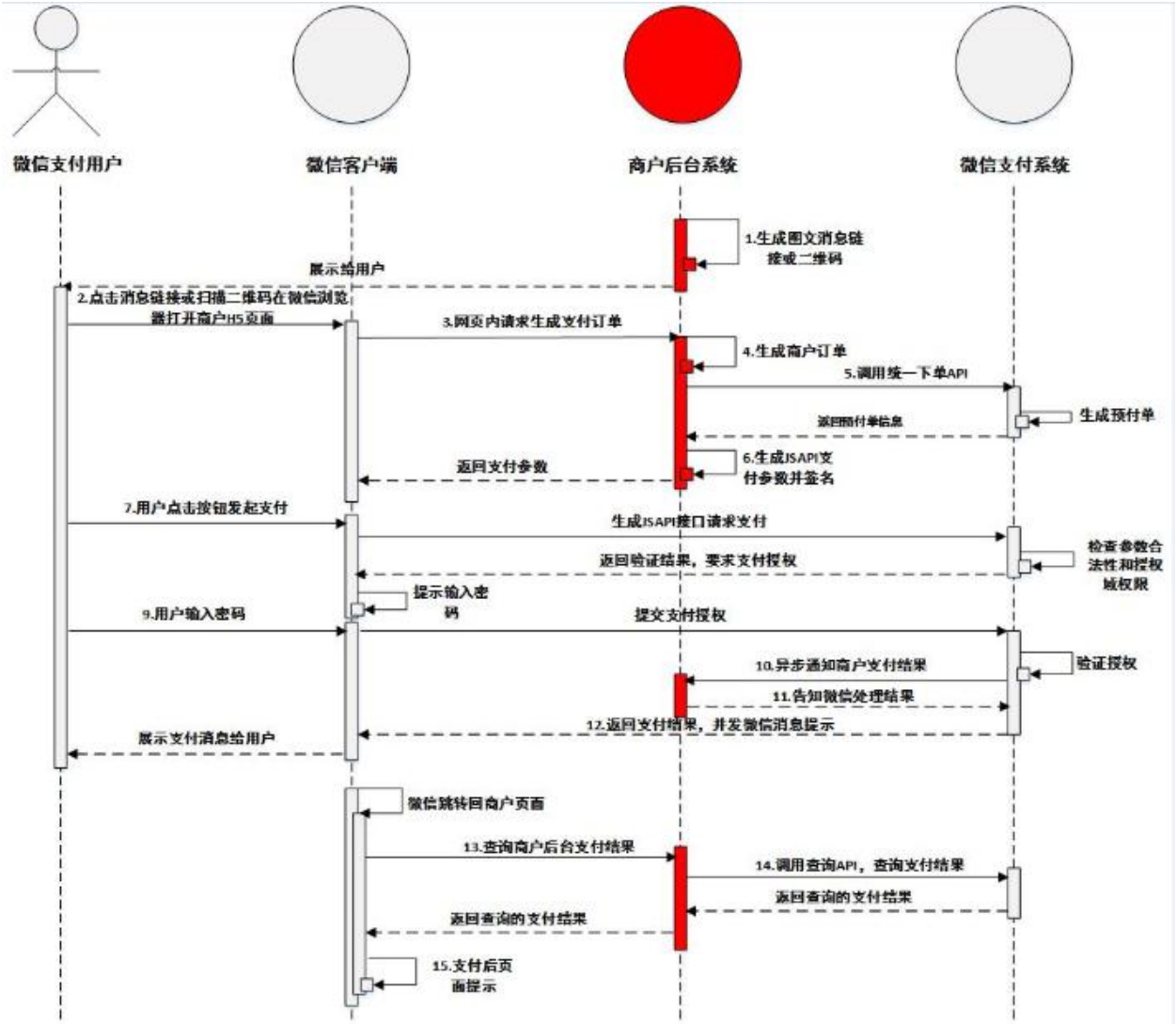

图 4-5 微信支付时序图

（6）微信红包

微信红包功能主要用于用户提现，当系统给用户提现时，系统会将微信红包请求微信服务器，请求成功后微信服务器返回微信红包是否发放成功的数据，系统将再次请求微信服务器，查询这些数据的真实性，若数据属实则写入数据库。流程图如图 4-6 所示：

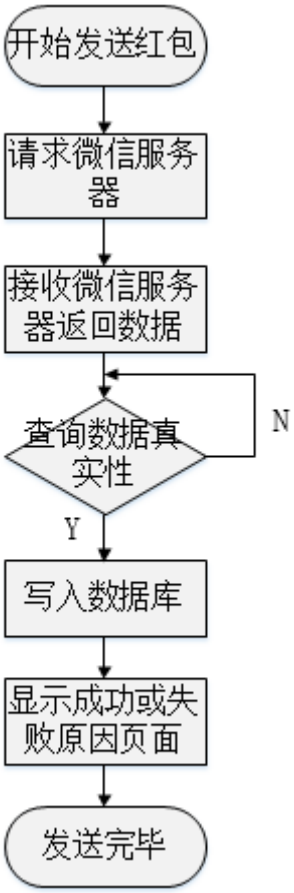

图 4-6 微信红包

4.2.2 微信公众号功能模块

微信公众号功能模块要实现的功能包括自定义菜单、素材上传、群发消息。下面分别设计这些功能：

（1）自定义菜单

用户在系统中对菜单进行增加、修改、删除和查看操作，确认菜单无误后点击“发布”按钮，系统将读取该微信公众号在系统数据库中的自定义菜单数据，将数据请求微信服务器，接收到微信服务器返回的数据后，将自定义菜单成功与否的结果显示给用户。流程图如图 4-7 所示：

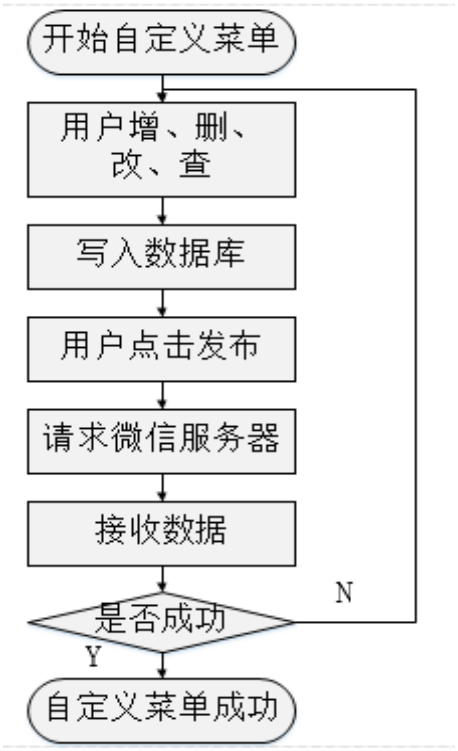

图 4-7 自定义菜单

（2）素材上传

用户在系统中上传素材，系统将用户上传的素材数据请求微信服务器，接收到微信服务器返回的数据后，判断上传是否成功，若成功则写入数据库并显示给用户，若不成功则直接将错误信息显示给用户。流程图如图 4-8 所示：

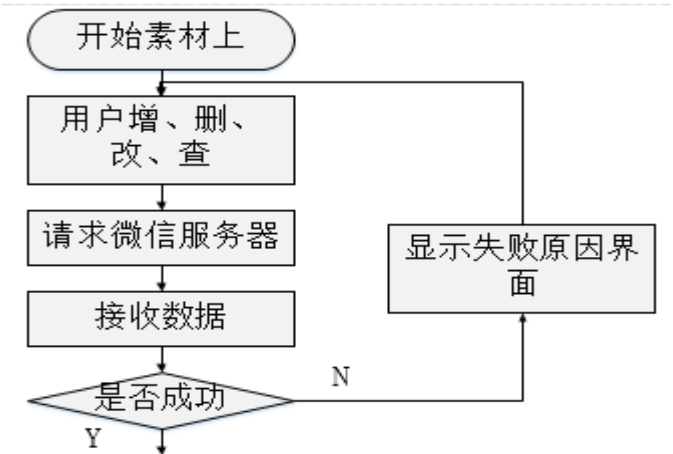

图 4-8 素材上传

（3）群发消息

用户在群发消息功能中，选择需要群发的素材，由于本系统目前仅仅支持群发给所有人，所以用户选择好素材后直接点击“发送”按钮，系统就会将数据请求微信服务器，接收到微信服务器返回的数据后，判断上传是否成功，若成功则写入数据库并显示给用户，若不成功则直接将错误信息显示给用户。流程图如图4-9 所示：

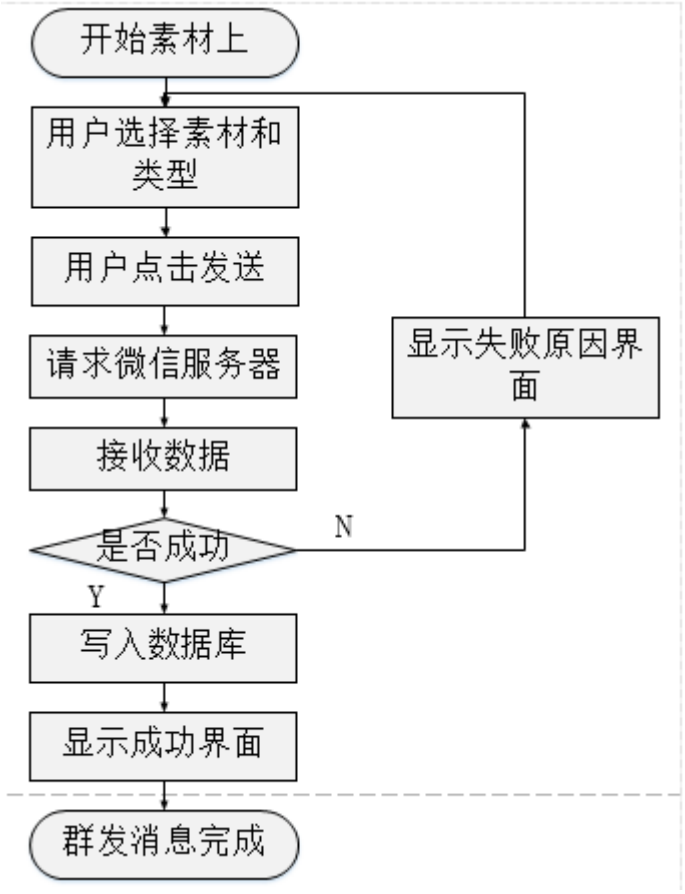

图 4-9 群发消息

4.3 数据库设计

（1）微信公众号表，表名：cmf_gl_wx_mp，结构如下表 4-1 所示。

表 4-1 微信公众号表

名称	类型	是否为空	默认值	备注
id	int(11)	主键	AUTO_INCREMENT	自动生成
org_id	int(11)	NO		组织 id
type	tinyint(1)	NO		公众号四种类型
name	char(50)	NO		公众号名称
wechat	char(100)	NO		拥有者的微信号
app_id	char(50)	NO		应用 ID
app_secret	char(50)	NO		应用密钥
ori_id	char(100)	NO		公众号原始 id
token	char(32)	YES	NULL	2-32 字符必须为英文或者数字
encoding_key	char(43)	YES	NULL	43 位加密字符
encoding_type	tinyint(1)	YES	0	0 明文（默认），1 兼容，2 安全
qrcode_id	int(11)	NO		微信公众号二维码 id
status	tinyint(1)	YES	0	0 无效，1 有效
createTime	int(11)	YES	0	创建时间
updateTime	int(11)	YES	0	更新时间

（2）微信公众号二维码表，表名：cmf_gl_wx_mp_qrcode，结构如下表 4-2 所示。

表 4-2 微信公众号二维码表

名称	类型	是否为空	默认值	备注
id	int(11)	主键	AUTO_INCREMENT	自动生成
uid	int(11)	NO		用户 id
mp_id	int(11)	NO		微信公众号 id
storageType	tinyint(4)	YES	0	存储类型，0 临时，1 本地，2 云存储
path	varchar(255)	NO		图片路径
create_time	int(11)	NO		创建时间

（3）微信公众号自定义菜单表，表名：cmf_gl_wx_mp_func_menu，结构如下表 4-3 所示。

表 4-3 微信公众号自定义菜单表

名称	类型	是否为空	默认值	备注
id	int(11)	主键	AUTO_INCREMENT	自动生成
sort	tinyint(4)	YES	0	排序号
pid	int(11)	YES	0	一级菜单
name	varchar(50)	NO		菜单名
keyword	varchar(100)	YES	NULL	关联关键词
url	varchar(255)	YES	NULL	关联 URL
mp_id	int(11)	NO		微信公众号 id
type	varchar(30)	YES	NULL	类型
from_type	tinyint(4)	YES	-1	配置动作
addon	char(50)	YES	0	选择插件
target_id	int(10)	YES	NULL	选择内容
sucai_type	varchar(50)	YES	NULL	素材类型
jump_type	char(10)	YES	0	推送类型

（4）微信公众号素材上传图片路径表，表名：cmf_gl_wx_mp_func_image，结构如下表 4-4 所示。

表 4-4 微信公众号素材上传图片路径表

名称	类型	是否为空	默认值	备注
id	int(11)	主键	AUTO_INCREMENT	自动生成
model	varchar(30)	YES	NULL	模型（去除 wxmp，如：MaterialNews）
used_id	int(11)	NO		使用者的 id
uid	int(11)	NO		发布操作者
mp_id	int(11)	NO		微信公众号 id
storageType	tinyint(4)	YES	0	存储类型，0 临时，1 本地，2 云存储
path	varchar(255)	NO		图片路径
create_time	int(11)	NO		创建时间

（5）微信公众号图片素材表，表名：cmf_gl_wx_mp_func_material_image，结构如下表 4-5 所示。

4-5 微信公众号图片素材表

名称	类型	是否为空	默认值	备注
id	int(11)	主键	AUTO_INCREMENT	自动生成
cover_id	int(11)	YES	0	图片在本地的 ID
media_id	varchar(100)	YES	0	微信端图文消息素材的 media_id
url	varchar(255)	YES	NULL	微信端的图片地址
mp_id	int(11)	NO		微信公众号 id
isCover	tinyint(4)	YES	0	是哪种素材:0 图文素材封面，1 图片素材

（6）微信公众号图文素材表，表名：cmf_gl_wx_mp_func_material_news，结构如下表 4-6 所示。

4-6 微信公众号图文素材表

名称	类型	是否为空	默认值	备注

id	int(11)	主键	AUTO_INCREMENT	自动生成
title	varchar(100)	YES	NULL	标题
author	varchar(30)	YES	NULL	作者
cover_id	int(11)	YES	0	封面
digest	varchar(255)	YES	NULL	摘要
content	text	YES	NULL	内容
link	varchar(255)	YES	NULL	外链
group_id	int(11)	YES	0	多图文组的 ID
thumb_media_id	varchar(100)	YES	NULL	图文消息的封面图片素材 id（必须是永久 mediaID）
media_id	varchar(100)	YES	0	微信端图文消息素材的 media_id
mp_id	int(11)	NO		微信公众号 id
create_time	int(11)	YES	0	发布时间
update_time	int(11)	YES	0	更新时间

（7）微信公众号群发消息表，表名：cmf_gl_wx_mp_func_message，结构如下表 4-7 所示。

4-7 微信公众号群发消息表

名称	类型	是否为空	默认值	备注
id	int(11)	主键	AUTO_INCREMENT	自动生成
uid	int(11)	NO		用户 id
mp_id	int(11)	NO		微信公众号 id
type	varchar(30)	YES	NULL	类型
material_id	int(11)	NO		素材 ID
errcode	varchar(30)	YES	NULL	错误码
errmsg	varchar(30)	YES	NULL	错误信息
msg_id	varchar(30)	YES	NULL	消息发送任务的 ID
msg_data_id	varchar(50)	YES	NULL	消息的数据 ID
create_time	int(11)	NO		创建时间

5 系统实现

5.1 功能模块实现

5.1.1 基本功能模块

微信基本功能模块要实现的功能包括配置微信公众号基本信息、微信公众号接入服务器、获取 accesstoken、获取用户基本信息、微信支付和微信红包。根据设计，下面分别是这些功能的部分实现核心代码以及前端界面图片：

（1）配置微信公众号基本信息

后端核心代码：

```
public function doSetting_1(){
$mp_id=I('get.mp_id');
$data=I('post.data');
$org_id=$data['org_id'];
$uid=$this->userId;
$this->requireLogin();
if($mp_id){
$wxMp=GlWxMpLogic::modify($uid,$mp_id,$data);
if($wxMp['ret']==1){
$this->success('修改成功
',sp_plugin_url('GlWxMp://Index/setting_2',array('mp_id'=>$mp_id,'id'
=>$org_id)));
}
else{
$this->error('修改失败');
}
}
else{
$res=GlWxMpLogic::checkRepeat($data['app_id']);
if($res['ret']==0){
$this->error('公众号重复');
}
$res=GlWxMpApiLogic::checkWxMpSetting($data);
```

```
if($res['ret']==0){
$this->error('公众号配置有错，请检查填写的配置');
}
$wxMp=GlWxMpLogic::create($uid,$org_id,$data);
if($wxMp['ret']==1){
$this->success('添加成功
',sp_plugin_url('GlWxMp://Index/setting_2',array('mp_id'=>$wxMp['data
'],'id'=>$org_id)));
}
else{
$this->error('添加失败');
}
}
}
```

前端界面如图 5-1：

图 5-1 设置微信公众号信息

（2）获取 accesstoken

后端核心代码：

```
public static function getAccessToken($wxMp){
```

```
$wxMp=self::getWechatParmeter($wxMp);
$wechat=new Wechat($wxMp);
$accessToken=$wechat->checkAuth();
if($accessToken){
return self::returnSuccess($accessToken);
}
return self::returnFail('获取 accessToken 失败');
}
```

（3）获取用户基本信息

后端核心代码：

```
$result=GlWxMpLogic::getDefaultWxMp();
if($result['ret']!=1){
$this->error('系统默认公众号数据获取失败');
exit;
}
$wxmp=$result['data'];
$result=GlWxMpApiLogic::getOpenId($wxmp);
if($result['ret']!=1){
$url='http://'.$_SERVER['HTTP_HOST'].'/api/plugin/execute/';
$pos=strpos($_SERVER["QUERY_STRING"], '_plugin');
$behindurl=substr($_SERVER["QUERY_STRING"],$pos);
$behindurl=str_replace('&','/',$behindurl);
$url.=str_replace('=','/',$behindurl);
GlWxMpApiLogic::oauthRedirect($wxmp,$url);
return;
}

public static function getOauthUserInfo($accessToken,$open_id){
$wechat=new Wechat();
$data=$wechat->getOauthUserinfo($accessToken,$open_id);
if($data){
return self::returnSuccess($data);
}
return self::returnFail('获取用户信息失败');
```

```
    }
```

（4）微信支付

后端核心代码：

```
    public static function showPayPage($payment,$wxMpOpenId=null){
    ini_set('date.timezone','Asia/Shanghai');
    $input = new \WxPayUnifiedOrder();
    $input->SetBody($payment['title']);
    $input->SetOut_trade_no('race'.$payment['id']);
    $input->SetTotal_fee($payment['money']*100);
    $input->SetNotify_url("http://clubs.91race.com/api/plugin/execute
/_plugin/GlWxMp/_controller/WePay/_action/notify");
    if($wxMpOpenId){
    //微信公众号支付
    $payType="JSAPI";
    $input->SetTrade_type($payType);
    $input->SetOpenid($wxMpOpenId);
    $order = \WxPayApi::unifiedOrder($input);
    $jsApiParameters = self::GetJsApiParameters($order);
    $return_data['assign_data']['payType']=$payType;
    $return_data['assign_data']['jsApiParameters']=$jsApiParameters;
    }
    else{
    //微信扫码支付.模式二
    $payType="NATIVE";
    $input->SetTrade_type($payType);
    $input->SetProduct_id($payment['id']);
    $result = \WxPayApi::unifiedOrder($input);
    $return_data['assign_data']['payType']=$payType;
    $return_data['assign_data']['code_url']=$result["code_url"];
    $return_data['assign_data']['check_status_url']=sp_plugin_url('Gl
Trade://Index/checkPaymentStatus',array('payment_id'=>$payment['id'])
);
    }
    $return_data['assign_data']['payment']=$payment;
```

```
return self::returnSuccess($return_data);
}
```

（5）微信红包

后端核心代码：

```
public static function sendCashBonus($data,$timeOut = 6){
$url =
"https://api.mch.weixin.qq.com/mmpaymkttransfers/sendredpack";
$wxMpCashBonus=new \WxMpCashBonus();
$wxMpCashBonus->setNonce_str($wxMpCashBonus->getNonceStr());
$wxMpCashBonus->setMch_billno($data['mch_billno']);
$wxMpCashBonus->setMch_id($data['mch_id']);
$wxMpCashBonus->setWxappid($data['appid']);
$wxMpCashBonus->setSend_name($data['send_name']);
$wxMpCashBonus->seTre_openid($data['openid']);
$wxMpCashBonus->setTotal_amount($data['total_amount']*100);
$wxMpCashBonus->setTotal_num($data['total_num']);
$wxMpCashBonus->setWishing($data['wishing']);
$wxMpCashBonus->setClient_ip($data['client_ip']);
$wxMpCashBonus->setAct_name($data['act_name']);
$wxMpCashBonus->setRemark($data['remark']);
$wxMpCashBonus->setSign();
$xml=$wxMpCashBonus->ToXml();
$response = $wxMpCashBonus->postXmlCurl($xml, $url, true, $timeOut);
$result = $wxMpCashBonus->FromXml($response);
return $result;
}
```

5.1.2 微信公众号功能模块

微信公众号功能模块要实现的功能包括自定义菜单、素材上传、群发消息。根据设计，下面分别是这些功能的部分实现核心代码以及前端界面图片：

（1）自定义菜单

后端核心代码：

```
public static function createMenu($wxMp,$data){
$wxMp=self::getWechatParmeter($wxMp);
```

```
    $wechat=new Wechat($wxMp);
    if($wechat->createMenu($data)){
    return self::returnSuccess();
    }
    return self::returnFail('errCode:'.$wechat->errCode.'
errMsg:'.$wechat->errMsg);
    }

    public static function deleteMenu($wxMp){
    $wxMp=self::getWechatParmeter($wxMp);
    $wechat=new Wechat($wxMp);
    if($wechat->deleteMenu()){
    return self::returnSuccess();
    }
    return self::returnFail('删除菜单失败');
    }

    public function doPost(){
    $data=I('data');
    if($data['pid']!=0 && $data['from_type']==0){
    $this->error('二级菜单的配置动作不能为空');
    }
    if($data['from_type']==0){
    unset($data['type']);
    }
    else{
    if(empty($data['type'])){
    $this->error('类型不能为空');
    }
    if($data['type']=='url' && empty($data['url'])){
    $this->error('跳转URL不能为空');
    }
    }
    if(!empty($data['url'])){
```

```
$data['url']=str_replace('%26','&',$data['url']);
}
if(empty($this->menu_id)){
//写入数据库
$data['mp_id']=$this->mp_id;
$result=GlWxMpFuncMenuLogic::add($data,$this->userId);
if($result['ret']!=1){
$this->error('添加菜单失败：'.$result['data']);
}
}
else{
$data['mp_id']=$this->mp_id;
$result=GlWxMpFuncMenuLogic::modify($data,$this->userId,$this->menu_id);
if($result['ret']!=1){
$this->error('修改菜单失败：'.$result['data']);
}
}
$this->success('菜单变更完成',sp_plugin_url('GlWxMpFunc://Menu/index',array('id'=>$this->org_id)));
}
```

前端界面如 5-2 图：

图 5-2 添加菜单

（2）素材上传

后端核心代码：

```
public static function uploadImg($wxMp,$data){
$wxMp=self::getWechatParmeter($wxMp);
$wechat=new Wechat($wxMp);
$result=$wechat->uploadImg($data);
if($result){
return self::returnSuccess($result);
}
return self::returnFail('errCode:'.$wechat->errCode.'
errMsg:'.$wechat->errMsg);
}

public static function
uploadForeverMedia($wxMp,$data,$type,$is_video=false,$video_info=arra
y()){
$wxMp=self::getWechatParmeter($wxMp);
$wechat=new Wechat($wxMp);
$result=$wechat->uploadForeverMedia($data,$type,$is_video,$video_
info);
if($result){
return self::returnSuccess($result);
}
return self::returnFail('errCode:'.$wechat->errCode.'
errMsg:'.$wechat->errMsg);
}

public static function uploadForeverArticles($wxMp,$data){
$wxMp=self::getWechatParmeter($wxMp);
$wechat=new Wechat($wxMp);
$result=$wechat->uploadForeverArticles($data);
if($result){
return self::returnSuccess($result);
}
```

```
    return self::returnFail('errCode:'.$wechat->errCode.'
errMsg:'.$wechat->errMsg);
    }
```

前端界面如图 5-3：

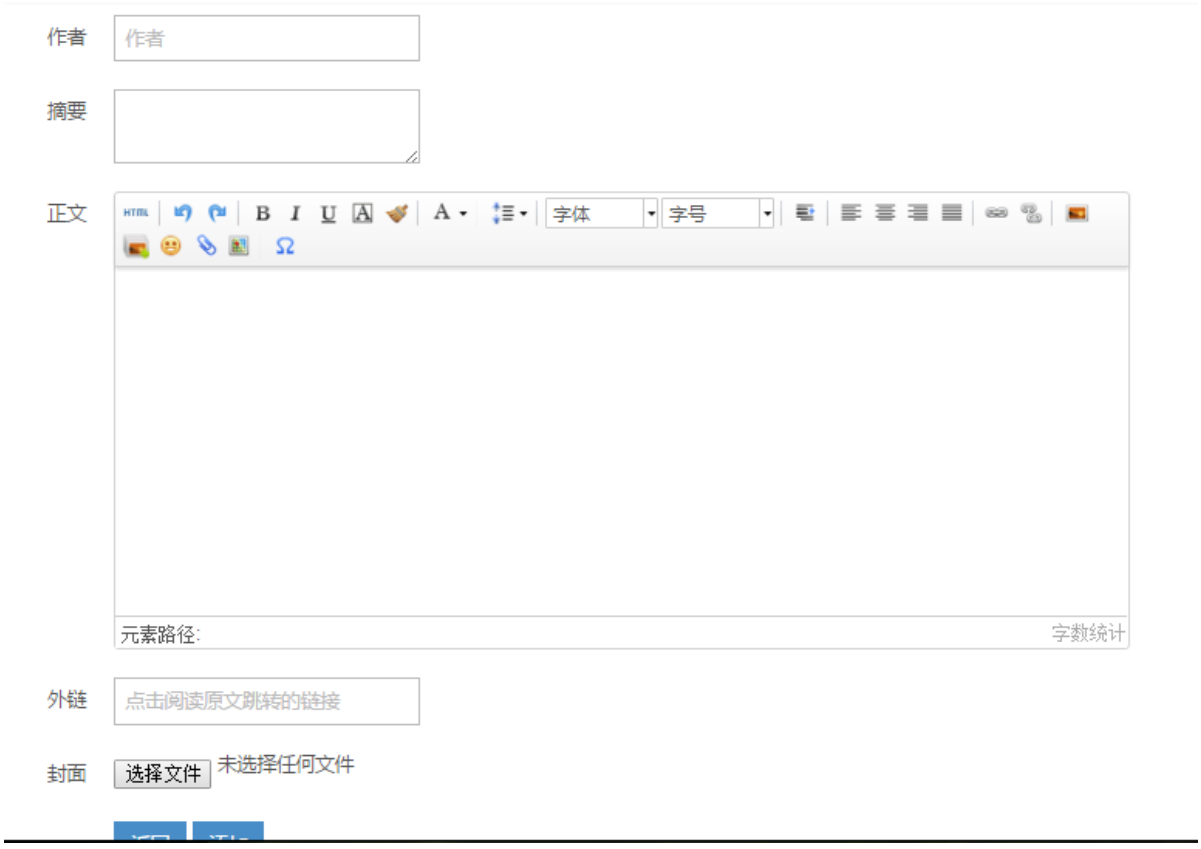

图 5-3 自定义菜单

（3）群发消息

后端核心代码：

```
public static function sendGroupMassMessage($wxMp,$data){
```

```
    $wxMp=self::getWechatParmeter($wxMp);
    $wechat=new Wechat($wxMp);
    $result=$wechat->sendGroupMassMessage($data);
    if($result){
    return self::returnSuccess($result);
    }
    return self::returnFail('errCode:'.$wechat->errCode.'
errMsg:'.$wechat->errMsg);
    }
```

前端界面如图 5-4：

图 5-4 消息管理

6 问题及解决方案

6.1 系统实现中遇到的问题及其解决方案

在系统实现过程中，难免会遇到许多问题而导致进度卡壳，在本系统的实现中本人遇到了许多问题，如：如何将微信公众号接口给其他模块调用、如何缓存各个微信公众号的 accesstoken 才是最好的等等。以下是本人罗列出每个遇到的问题及其解决方案。

（1）如何将微信公众号接口给其他模块调用

由于官方给出的 SDK 只有每个接口的代码，没有类可言，调用起来比较不方便，所以本人下载了别人封装好的 SDK。但是如果直接使用，在一些数据处理上的代码就没办法复用，由于插件的结构有个 Logic，所以本人将使用到的功能封装成一个静态的 ApiLogic 类，还将数据处理的代码封装成一个方法，当其他模块需要调用微信公众号接口时，只需要引用这个 ApiLogic，直接使用类名::方法名调用即可。

（2）验证是否是组织管理员、获取参数等代码如何在代码结构上复用

在实现微信公众号功能模块时，本人发现每个 controller 中的每一个方法几乎都需要验证登录者是否是组织的管理员，由于本人将自定义菜单、素材上传、群发消息各分为一个 controller，他们的每一个方法几乎都需要获取他们各自的 ID，比如自定义菜单编辑和删除时需要 get 一个自定义菜单 ID，这些代码都是可以复用的。这个问题的解决方案是本人在微信公众号模块的 controller 中实现一个这个模块的基类，这个基类的_initialize（初始化）方法中验证是否是组织的管理员，每个功能的 controller 继承这个基类，并在自己的 _initialize 方法中实现父类的_initialize，并且获取本功能的 ID。这样每个方法中就不需要实现验证是否是管理员和获取参数的代码了。

（3）如何缓存各个微信公众号的 accesstoken 才是最好的

由于本人使用的 SDK 中没有实现对 accesstoken 的缓存，只有申明缓存的方法，所以本人使用 S 方法（application 缓存），用各个微信公众号的应用 ID（appid）作为缓存名称，将各个微信公众号的 accesstoken 缓存起来。

（4）图文素材上传时，内容中包含图片的上传算法

由于图文素材上传时，内容中若包含图片，图片的链接不能是除了微信服务器以外的链接，必须将每个图片通过特定的上传接口，将图片上传到微信服务器，获取在微信服务器中的链接。本人使用的算法是将每个 img 标签中的 src 属性的

值取出，然后通过接口上传图片获取链接再将值替换，但是在实现的过程中遇到不少麻烦，最后的实现代码如下：

```
private function getNewContent($content)
{
if (! $content)
return;
$newUrl = array();
preg_match_all('#<img.*?src="([^"]*)"[^>]*>#i', $content, $match);
foreach ($match[1] as $mm) {
$mms['media']='@'.realpath(SITE_PATH.substr($mm,strpos($mm,'/data')));
$newUrl[$mm] = $this->addArtContentImage($mms);
$newUrl[$mm]=$newUrl[$mm]['data']['url'];
}
if (count($newUrl)){
$content_new = strtr($content, $newUrl);
}
return empty($content_new) ? $content : $content_new;
}
```

（5）群发消息时微信返回成功却没有群发出去

这个问题需要注意以下几点：

1）有可能是因为网络延迟没有群发出去，但是如果过了很久都没有收到，那就不是这个原因。

2）有可能是因为素材内容的字符太多，官方的文档中说明了字符不能超过 2 万，但是本人实际的是超过 4 万就一定群发不出去。

3）在实际的测试过程中，本人确定了网络没有延迟，字符数没有超过，此时本人怀疑是某些字的原因使整个消息被微信屏蔽，具体的原因还不明确，在将最后的一些字删除之后群发又是可以的，所以如果群发不出去，还要做下这个测试。

（6）自定义菜单和群发消息中都存在链接，如果链接中带&符号，前端发送给后端时这个链接会在&后面多出‘amp;’，导致链接变得不可用。

这个问题的原因是因为&是转义字符，当前端发送给后端一个带&链接时就会在这个转义字符后面多出‘amp;’，这个问题的解决方案是在前端发送给后端数据时，通过 javascript 将链接中的&字符转换成‘%26’，’ %26‘在 URL 中也代表着&字符，当后端收到数据后，再将‘%26’转换成&字符即可。具体的实现代码如下：

前端的 javascript 代码：

```
function urlencode(){
```

```
var url=$('#type_url').val();
if(url!=null){
url=url.replace(/\&/g,'%26');
$('#type_url').val(url);
}
$('#form').sub();
}
```

后端的 php 代码：

```
$data['url']=str_replace('%26','&',$data['url']);
```

结论

PHP 是完全免费的，它和 MySQL 及 Apache 的配合使用已经被许多网站的设计人员所采用[7]。而 ThinkPHP 可以非常方便和快捷地开发和部署应用。不仅仅是在企业级应用级别，任何 PHP 应用开发都可以从 ThinkPHP 的简单和快速的特性中受益。ThinkPHP 本身具有很多的原创特性，并且倡导代码至简，用最少的代码完成更多的功能，宗旨就是让 Web 应用开发更简单、更快速。

在互联网技术不断更新的今天,企业的营销方式也在不断向前发展。以微信为代表的互联网平台成了企业重要的营销工具,而微信公众平台作为带有社会化媒体属性和虚拟社区发展条件的新型互联网平台,在其中扮演着重要的角色[8]。

目前，网络中的微信公众号服务系统有非常多，有开源的，也有收费的，但是大多数都是作为一个大系统中的一个功能使用。本课题所设计并实现的微信公众号亦是如此，它是作为基于社群的泛电商系统中的插件实现的。

虽然网络上的微信公众号服务系统繁多，但是各有其自身的优点。本系统的一个优点是由于本身就是基于一个 ThinkCMF 系统框架进行开发的，再加上对于 MVC 架构本人还做了一个 Logic 层次，使得整个系统的耦合度变得更低，所以整个系统架构非常的清晰，即便将整个系统作为开源系统，上手进行二次开发的速度是非常快的。但是由于本系统属于敏捷开发，大部分的前端页面的排版都未完成。

参与基于社群的泛电商系统项目的敏捷开发，第一个让本人感受到的是敏捷开发的速度。在开发过程中项目组先把关键路径上的部分实现，然后进一步推进，直到达到一个目前所需要的里程碑并且系统已经可以上线使用后，然后再回过头将不在关键路径上的部分实现，所以敏捷开发对于一个时间要求比较紧凑的项目开发是非常重要的。

在设计阶段，项目组提了一个 Logic 层次，在系统原本的 MVC 架构上加上 Logic，使得整个系统的耦合度降低，也降低了后期运维的成本和难度。

PHP is a popular language for server-side applications[9]. （PHP 是一种流行的服务器端应用程序的语言。）目前 PHP 工程师的需求量是非常大的，所以学好 PHP 语言对于本人未来有极大的帮助，且本人也对 PHP 语言非常有兴趣。

在大学时期，参与了一个大项目的开发，让本人学到了敏捷开发的过程，也让本人学到了 PHP 技术，PHP 是一种服务器端、跨平台、HTML 嵌入式的脚本语言[10]，对于以后参加工作打下了一个基础。从只会写 C 代码的本人到学会写 PHP 代码的本人，整个开发过程还不仅仅学到的是 PHP 语言，还学到了 html、javascript、css 语言，受益良多。

参考文献

[1] 王俊芳,李隐峰,王池.基于 MVC 模式的 ThinkPHP 框架研究[J].电子科技,2014(4):151-153.

[2] 黄建.基于ORM的PHP框架研究与应用[D].西安:西安建筑科技大学计算机软件与理论专业,2010.

[3] 李慧,高飞.PHP 入门经典[M].北京:机械工业出版社,2013:398-399.

[4] 徐诚斌,王金平.MVC 在 ThinkPHP 框架中的应用研究[J].信息与电脑(理论版),2011(03):160-161.

[5] 开源中国.内容管理框架 ThinkCMF[EB/OL].[2013-12-19]. http://www.oschina.net/p/thinkcmf.

[6] 梁建平.常规高空风气象数据整编系统的研究与实现[D].成都:电子科技大学软件工程专业,2015.

[7] 邹天思,潘凯华,刘中华,等.PHP 数据库系统开发完全手册[M].北京:人民邮电出版社,2007:3-4.

[8] 程国民.微信公众平台用户参与行为影响因素研究[D].重庆:西南政法大学企业管理专业,2015.

[9] Akihiko Tozawa,Michiaki Tatsubori,Tamiya Onodera,Yasuhiko Minamide. Copy-on-write in the PHP language[J].ACM SIGPLAN Notices,2009,44 (1):200-212.

[10] 肖维明.基于 PHP+MySQL 的网站开发[J].物流工程与管理,2009(06):90-92.