Pig Group
软件设计文档

项目组组员

姓名(方向)	学号
梁子仲（嵌软）	14331153
林书群（计应）	14331162
刘劲（数媒）	14331176
陈向妍（计应）	14331038
王思佳（电政）	14331263

目录

1 引言

1.1 编写目的

编写软件设计文档是软件开发过程中必不可少的部分，其目的是为了使开发人员完成具体功能模块的具体实现的设计工作。具体而言，编写软件技术文档的目的是为所开发的软件提供以下功能：

- 设计系统方案。首先把复杂的功能进一步分解成简单的功能，遵循模块划分独立性原则，使划分过的模块的功能对大多数程序员而言都是易懂的。功能的分解导致对数据流图的进一步细分，并选用相应图形工具来描述。
- 提供一组合理的方案，并准备好系统流程图，系统的物理元素清单，实现系统的进度计划；
- 推荐最佳实施方案综合分析对比各种合理方案的利弊，推荐一个最佳的方案，并为最佳方案制定详细的实现计划。

1.2 项目技术选型及理由

前端：Html+CSS+JQuery。简单，灵活的 HTML，CSS，JQuery 框架，用于流行的用户界面组件和交互。

后端：Python Flask。选取理由是因为 Flask 是一个用 Python 编写的轻量级的 Web 应用框架，它使用简单的核心，可以用 extension 增加其他功能，自带开发用服务器和 debugger。

数据库：MySQL。MySQL 是最流行的关系型数据库管理系统，在 Web 应用方面 MySQL 是最好的 RDBMS（Relational Database Management System，关系数据库管理系统）应用软件之一。由于其体积小、速度快、总体拥有成本低等特点，本网站开发选择 MySQL 作为网站数据库。

渲染页面：Bootstrap。因为它是目前最受欢迎的前端框架，简洁灵活，可以使得 Web 开发更加快捷。

项目构建：GitHub。在 GitHub 上创建小组的开源项目并与组员们协作编

码，可以使得小组成员为项目贡献代码非常简易。

2 模块划分

本网站的模块可分为六个，分别是配置和运行，发送验证信件，数据库模型，表单和路由，页面及 css/js，虚拟环境和数据库迁移。具体模块划分图如下所示。

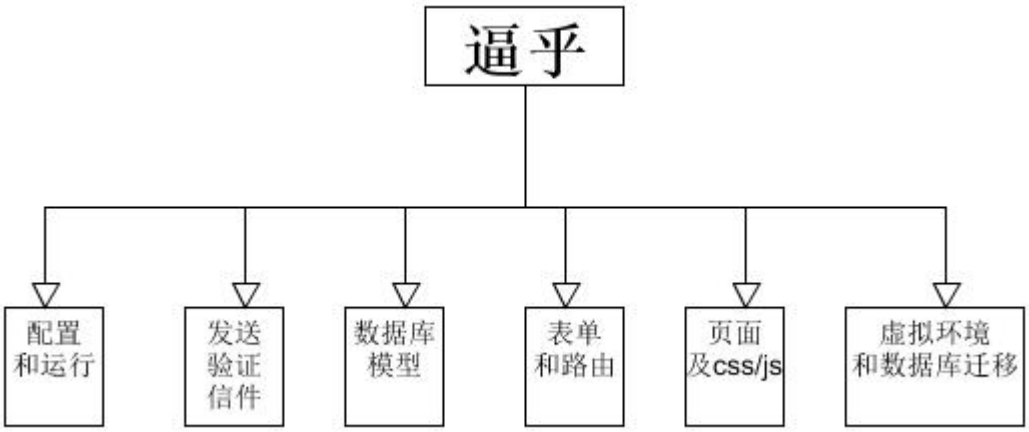


图 2.1 模块划分图

模块接口设计如下：

配置和运行模块将其余模块整合并调用起来，达到开启服务的目标

表单和路由模块调用数据库模型，获取或修改相关数据库数据

表单和路由模块渲染页面并进行展示

页面将调用相对应的 css 和 js 进行表示

发送信件会在表单路由中的注册以及修改信息过程中调用

数据库迁移以及虚拟环境的配置确保服务器正常运行，数据库正常使用

3 架构设计

3.1 网站平台架构

本网站平台采用 MVC（Model-View-Controller）架构模式。MVC 模式是

软件工程中一种常用的软件架构模式，它把软件系统分为三个基本部分：模型（Model）、视图(View)、和控制器(controller)。

模型（Model）用于封装与应用程序的业务逻辑相关的数据以及对数据的处理方法。模型有对数据具有直接访问权力，且模型不依赖视图和控制器。模型中的数据变化一般会通过一种刷新机制被公布。

视图(View)能够实现数据的显示。为了实现视图上的刷新功能，视图需要访问它监视的数据模型，因此视图应在数据这一部分注册，以便收到通知。

控制器(controller)接收用户的请求，并调用合适的资源来执行请求。它处理事件并做出响应，事件包括用户的行为和数据模型上的改变。

本网站的 MVC 架构设计如下图所示。

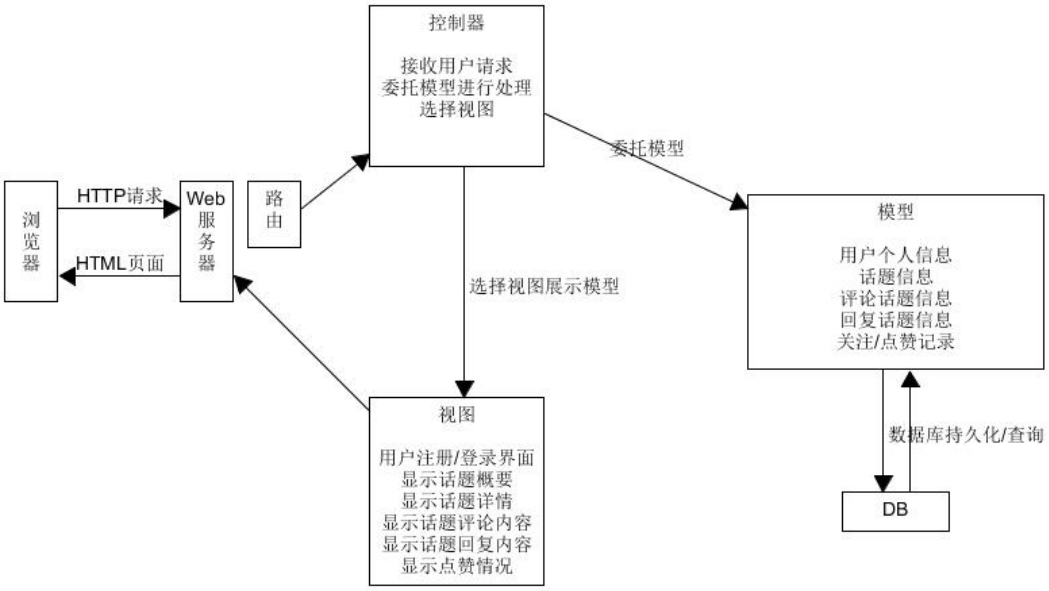
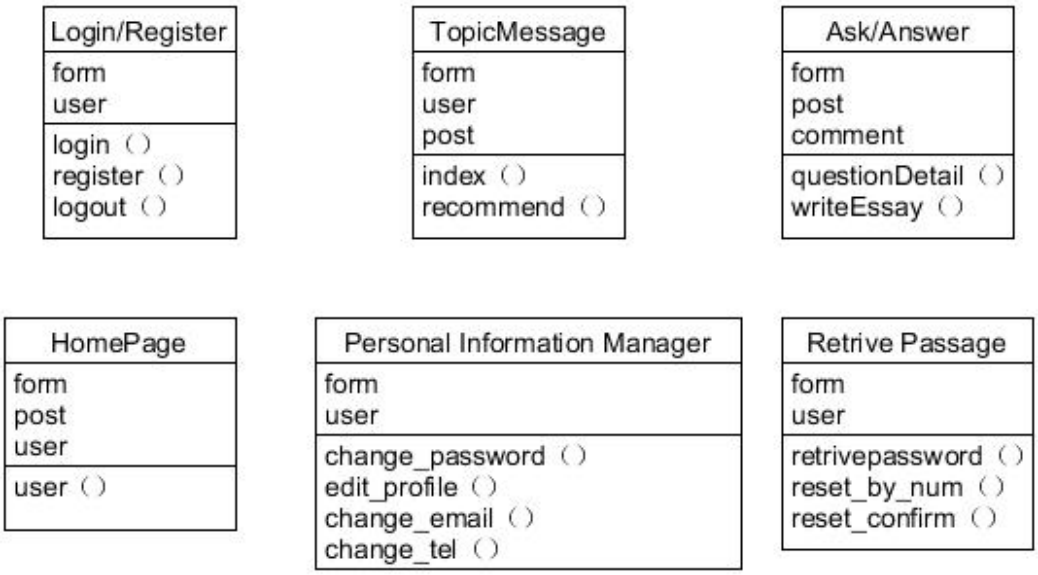


图 3.1 网站的 MVC 架构模型

3.2 关键抽象类

经过详细的分析和讨论，本平台所采用的类主要有 6 个，分别是登录/注册类、主页及推荐话题类、话题/评论信息类、个人主页类、修改个人信息类、密码找回类。总的类图最终如下所示。



4 用例分析

每个用例分析由两部分组成，第 1 部分用例功能描述，对用例功能进行简单的描述；第 2 部分用例交互过程，主要描述了用户与网站系统的交互工程。

4.1 注册/登录用例分析

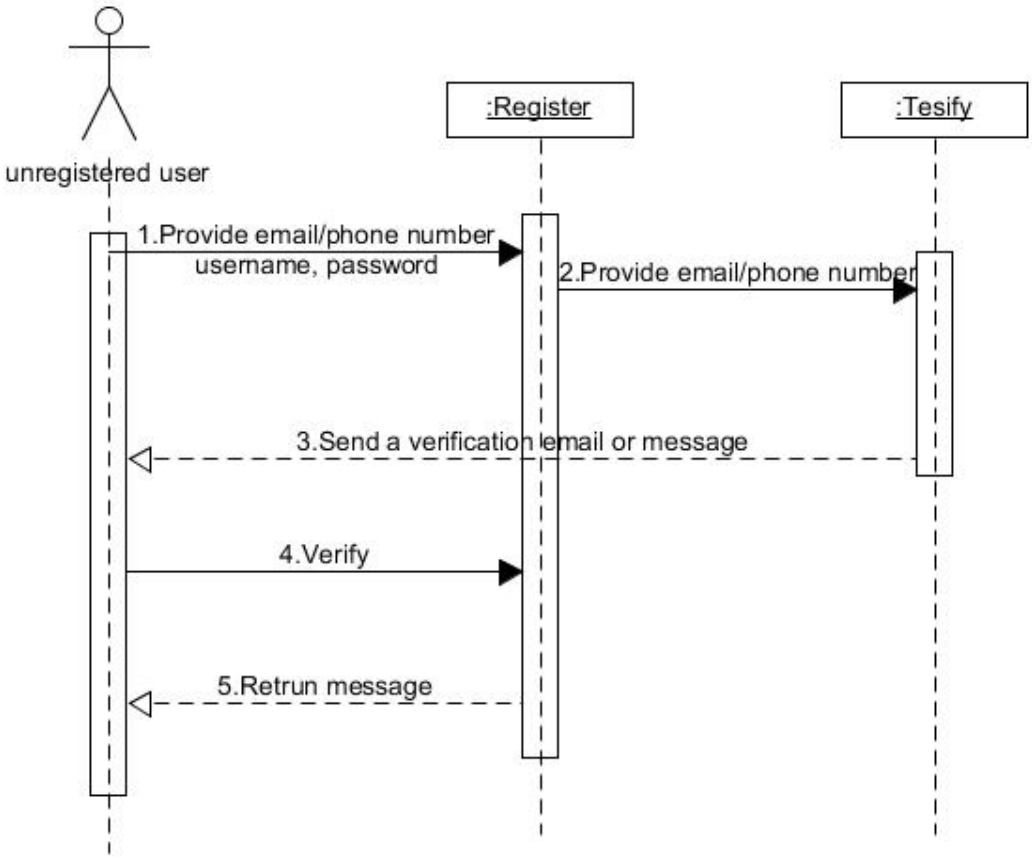
4.1.1 注册/登录用例功能描述

注册/登录用例由三个子事件组成，分别是邮箱/手机注册，用户登录，密码找回。这里选取子事件邮箱/手机注册做详细描述，其他子事件与它相似，就不在一一描述。邮箱/手机注册用例功能描述为：用户通过有效邮箱或手机进行注册，通过邮箱/手机的验证后，可从匿名用户转变为注册用户。

4.1.2 注册/登录用例交互过程

- 用户进入注册界面；

- 用户提供注册所用有效邮箱账号或手机号码，以及设置用户名和登录密码；
- 系统发送验证邮件或短信到用户对应的账号；
- 用户通过验证，成为注册用户。



4.2 浏览话题信息用例分析

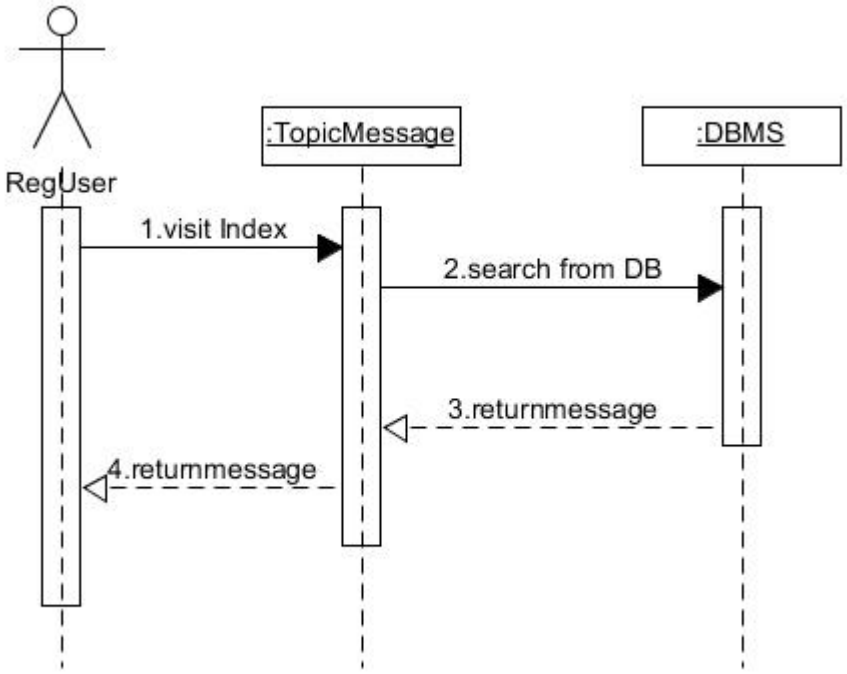
4.2.1 浏览话题信息用例功能描述

浏览话题信息用例允许不同权限的用户（匿名用户和注册用户）浏览各自权限之内的各种话题信息，包括话题概要信息、话题详细信息、评论内容、回复内容、关注/点赞情况等等。这里选取了浏览话题概要信息子事件做详细描述，其他子事件与此内容相似，就不再一一描述。浏览话题概要信息的功能描述为：用户进入网站浏览网站上发布的各种话题概要信息，包括所有的话题和系统推荐的

话题。

4.2.2 浏览话题信息用例交互过程

- 用户进入主页界面或话题推荐界面；
- 系统从话题目录系统中得到各类话题标题列表及话题概要信息，将话题标题按发布的先后顺序展示给用户，新发布的话题在前面；
- 用户从话题列表中浏览自己想了解的话题的概要信息。



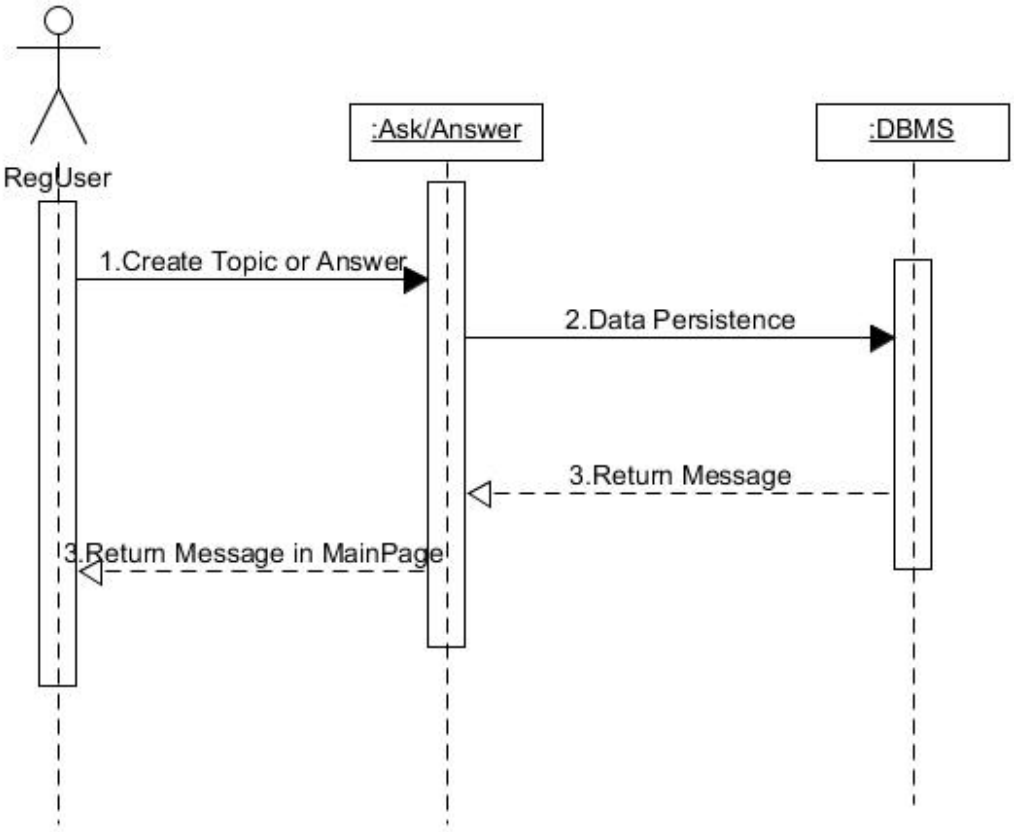
4.3 发表话题/评论/回复用例分析

4.3.1 发表话题/评论/回复用例功能描述

发表话题/评论/回复用例由三个子事件组成，这里选取发表话题子事件做详细描述，其他子事件与发布内容相似，就不在一一描述。发表话题的功能描述为：用户发布自己的疑惑或令人感兴趣的话题内容。

4.3.2 发表话题/评论/回复用例交互过程

- 用户进入添加话题界面，选择发表话题；
- 用户进入发表话题信息编辑界面，输入发表的内容，并点击提交；
- 发布界面逻辑层取得发布内容后，调用发布内容业务逻辑，进行发布内容逻辑的处理后，将发布内容持久化
- 成功持久化发布内容后返回发布成功确认提示，否则，提示用户发布失败。



4.4 个人信息管理用例分析

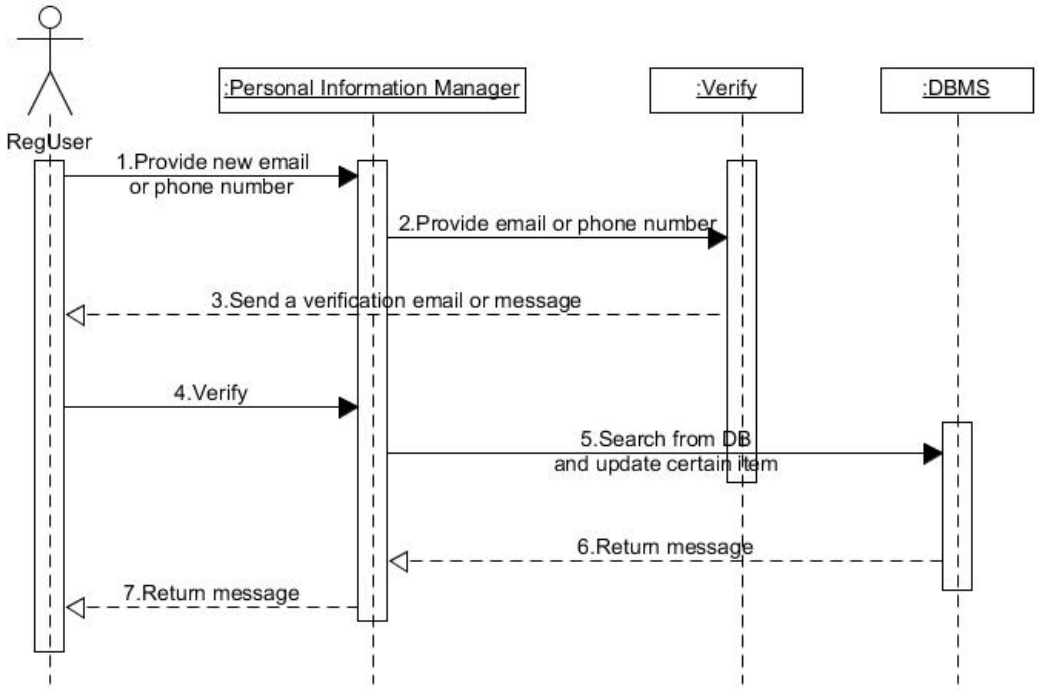
4.4.1 个人信息管理用例功能描述

个人信息管理用例包含四个子事件，这里选取修改邮箱子事件做详细描述，其他子事件与发布内容相似，就不在一一描述。修改邮箱的功能描述为：用户用

新邮箱取代曾用邮箱。

4.4.2 个人信息管理用例交互过程

- 用户进入个人信息界面，选择修改邮箱；
- 用户提供新的有效邮箱账号；
- 系统发送验证邮件到用户对应的账号；
- 用户通过验证，成功修改用户绑定的邮箱。

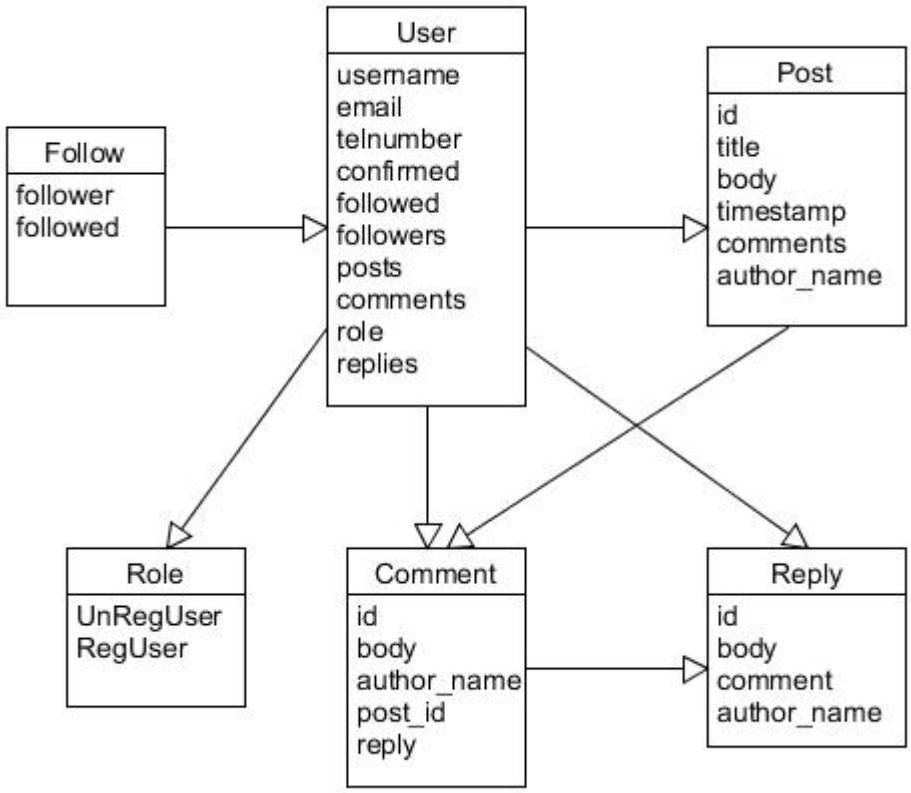


5 系统运行时架构设计

5.1 分析本网站系统的并发需求

经过本项目小组内部的讨论和分析，本系统没有并发需求，也没有事件驱动，没有复杂计算，因此本系统不需进行运行时架构设计。

6 数据库模型



7 设计技术

在本网站开发过程中用到的软件设计技术有：Structure Programming,Object-Oriented Programming，
Service Oriented Architecture,Design Patterns 等。

- Structure Programming 是一种编程典范，它采用子程序、程式代码区、for 循环以及 while 循环等结构，来取代传统的 goto，希望借此来改善程序的明晰性、品质以及开发时间，并且比 main 被写出面条式代码。

  开发中使用到 Structure Programming 的两个例子：

  retrive.js

```
if (password_confirm.length < 6) {
    iscorrect = false;
    $("#errorPasswordConfirm").css("visibility", "visible");
}

if (password !== password_confirm) {
    iscorrect = false;
    $("#errorPasswordConfirm").css("visibility", "visible");
    $("#errorPasswordConfirm").html('密码不一致');
}
```

jquery.js

```
for ( ; i < length; i++ ) {
    // Only deal with non-null/undefined values
    if ( (options = arguments[ i ]) != null ) {
        // Extend the base object
        for ( name in options ) {
            src = target[ name ];
            copy = options[ name ];

            // Prevent never-ending loop
            if ( target === copy ) {
                continue;
            }

            // Recurse if we're merging plain objects or arrays
            if ( deep && copy && ( jQuery.isPlainObject(copy) || (copyIsArray = jQuery.isArray(copy)) ) ) {
                if ( copyIsArray ) {
                    copyIsArray = false;
                    clone = src && jQuery.isArray(src) ? src : [];

                } else {
                    clone = src && jQuery.isPlainObject(src) ? src : {};
                }

                // Never move original objects, clone them
                target[ name ] = jQuery.extend( deep, clone, copy );

            // Don't bring in undefined values
            } else if ( copy !== undefined ) {
                target[ name ] = copy;
            }
        }
    }
}
```

总的来说，开发全程使用 Structure Programming，因此每一个模块几乎都可以找到 Structure Programming 的实践情况。

- Object-Oriented Programming 是一种计算机编程架构，它的基本原则是计算机程序是由单个能够起到子程序作用的单元或对象的组合而成。由于抽象性、封装型、重用性以及方便使用等方面的原因，以组件为基础的编程在脚本语言中已经变得特别流行。而 Python 语言在开发时完全采用 OOP 的思想，因此几乎每一个模块都可以找到 Object-Oriented Programming 的实践情况。
- Service Oriented Architecture（SOA）是一种粗粒度、松耦合服务架构，服务之间通过简单、精确定义接口进行通讯，不涉及底层编程接口和通讯模型。

在“发送验证邮件/短信”模块中使用到 SOA 技术，用异步的方式发送邮件，

使响应速度更快。

```
def send_async_email(app, msg):
    with app.app_context():
        mail.send(msg)

def send_mail(to, subject, template, **kwargs):
    app = current_app._get_current_object()
    msg = Message(app.config['MAIL_SUBJECT_PREFIX'] + ' ' + subject,
                  sender=app.config['MAIL_SENDER'], recipients=[to])
    msg.body=render_template(template+'.txt', **kwargs)
    msg.html=render_template(template+'.html', **kwargs)
    thr = Thread(target=send_async_email, args=[app, msg])
    thr.start()
    return thr
```

- Design Patterns

在设计模式方面，开发中主要用到原型模式（Prototype），即用原型实例指定创建对象的种类，并且通过拷贝这个原型来创建新的对象。

在“数据库模型”模块中进行定义：

```
class User(UserMixin, db.Model):
    __tablename__ = 'users'
    '''id = db.Column(db.Integer, primary_key=True)'''
    username = db.Column(db.String(64), unique=True, index=True, primary_ke
    email = db.Column(db.String(64), unique=True, index=True)
    password_hash = db.Column(db.String(128))
    telnumber = db.Column(db.String(16), unique=True, index=True)
    avatar_url = db.Column(db.String(128), default="avatar/head.png")
    user_detail = db.Column(db.Text())
    member_seen = db.Column(db.DateTime(), default=datetime.utcnow)
    last_seen = db.Column(db.DateTime(), default=datetime.utcnow)
    confirmed = db.Column(db.Boolean, default=False)
    role_id = db.Column(db.Integer, db.ForeignKey('roles.id'))
    posts = db.relationship('Post', backref='author', lazy='dynamic')
```

在“表单和路由”模块中的路由处理函数中进行实例化：

```
u = User(username=username, email=email, password=password)
db.session.add(u)
db.session.commit()
```