|  |
| --- |
| 中山大学软件学院2012级 |
| 软件设计文档 |
| ——公益活动管理系统Philosopher |
|  |
|  |
| **2015-7-18** |
|  |

**项目小组成员**

**12330054 陈跃东**

**12330053 陈远宏**

**12330051 陈宇文**

**12330084 封伟城**

**12330293田 澍**

目录

[一． 软件需求概述 3](#_Toc425456149)

[二． 技术选型说明 3](#_Toc425456150)

[1. 版本控制工具与团队管理 3](#_Toc425456151)

[2. 后端开发框架：Django on Python 4](#_Toc425456152)

[3. 数据库：SQLite 4](#_Toc425456153)

[4. 前端框架：Bootstrap、JQuery 4](#_Toc425456154)

[三． 架构设计 5](#_Toc425456155)

[四． 模块划分 6](#_Toc425456156)

[1. Model 6](#_Toc425456157)

[2. View 6](#_Toc425456158)

[3. Templat 6](#_Toc425456159)

[五． 技术实现 7](#_Toc425456160)

[1. Structured programming 7](#_Toc425456161)

[2. Aspect-Oriented Programming 8](#_Toc425456162)

[3. Object-oriented programming 9](#_Toc425456163)

[4. MVC 12](#_Toc425456164)

[六． 网站测试 12](#_Toc425456165)

[1. 功能测试 12](#_Toc425456166)

[2. 负载和压力测试 16](#_Toc425456167)

## 软件需求概述

鉴于当前许多同学因为公益时不够而发愁，主要的原因是没有一个好的公益活动信息平台。而我们的“公益活动发布平台”是一个B/S架构的平台，使得公益活动组织可方可以发布和管理公益活动，个人用户可以通过我们的平台参加相关公益活动。这样既便利了公益活动组织方对于活动的宣传和组织，同时也让同学们有了一个公益活动信息的“集散地”，能够及时得到各种公益活动信息。

每个公益活动组织方的注册都需要进行申请，并且经过系统管理员的同意。每个发布的公益活动都需要有相关信息，例如活动名称，公益时数，报名方式等等。同时，对于希望参加活动的申请者，公益活动组织方可以对他们上传的申请资料进行审核。若通过审核，则可通过系统向他们发送活动参与通知；反之，也可以向他们发送申请失败的通知。当公益活动结束后，公益活动组织方根据每个活动参与者的勤情况，确认活动参与者是否成功完成活动。若成功完成，系统会自动给每个成功参与者的账户加上相应的公益时。

而对于每个个人用户，都有一个与自己相关的活动列表，每个活动有一个状态，；例如：“正在申请”、“被拒绝”、“成功参与”。用户登录后可以浏览并申请相应的公益活动，。如果申请被通过，用户可以收到活动的参与通知。当活动结束，如果用户成功参与，活动组织方将进行确认，这样用户的公益时数将会相应增加，同时用户也可以对活动进行反馈。当然每个学期开始，用户的公益时数会进行清零操作。

综上所述，我们想构建一个轻量级，且功能完备，使用便利的在校公益发布平台。平台最终的实现形式将会是，基于PC上的B/S架构的网站系统。

## 技术选型说明

1. 版本控制工具与团队管理：Git、Github

* 由于是多人协同开发，所以为了进行团队内代码版本的控制，我们需要一个版本控制工具，这里我们选择了Git来作为我们团队的版本控制工具，Git是一个开源的分布式版本控制系统，用以有效、高速的处理从很小到非常大的项目版本管理。相比SVN来讲，它更适合团队开发，因为每个团队成员都可以保存一个独立的版本库，而SVN则只有一个中心版本库，且值得一提的是Git 是 Linus Torvalds 为了帮助管理 Linux 内核开发而开发的一个开放源码的版本控制软件，所以我们选择Git作为我们的版本控制工具
* 至于团队管理平台，我们选择Github，GitHub可以托管各种git库，并提供一个web界面，但与其它像 SourceForge或Google Code这样的服务不同，GitHub的独特卖点在于从另外一个项目进行分支的简易性。为一个项目贡献代码非常简单：首先点击项目站点的“fork”的按 钮，然后将代码检出并将修改加入到刚才分出的代码库中，最后通过内建的“pull request”机制向项目负责人申请代码合并。Github是全球最大的IT开源社区，许多著名的开源项目都在上做团队开发，例如Ruby on Rails和redis等。

### 后端开发框架：Django on Python

* Python作为一门脚本语言，虽然其执行效率上不及C与C++，但其最大的优势就是Python有一个庞大的模块系统，能提供许多功能强大的模块，基本上你能想到的她都有，用Python进行开发可以大大缩短开发周期和代码量，这是C与C++所不及的地方。所以为了缩短我们的开发周期，进行敏捷开发，我们选择Python作为我们的后端开发语言
* Django诞生于新闻网站的环境中，所以它提供很多了特性（例如她的的管理后台），非常适合内容类的网站，而我们所要开发的系统就是一个内容类的网站，所以在功能特性上刚好能够匹配
* Django是一个要求用户基于MVC设计模式进行开发的后端框架，易于整体设计，开发起来便捷而高效，这与我们构建一个功能完备的轻量级系统的想法不谋而合
* Django有活跃的社区支持，作为一个成熟的后端框架Django有丰富的技术文档作为参考，所以我们开发起来在技术上就少了很多后顾之忧

### 数据库：SQLite

在数据库的选择上我们就比较保守了，这里我们选择的是轻量级数据库SQLite，原因主要是我们所要构建的是一个轻量级的在校公益平台发布系统，所以在数据处理的并发性以及数据容量上并不会有太高要求，一个轻量级的SQLite就足以满足我们的需要了，而且SQLite是Django自带的，有方便的数据操作接口，也省去了环境搭建所需的时间

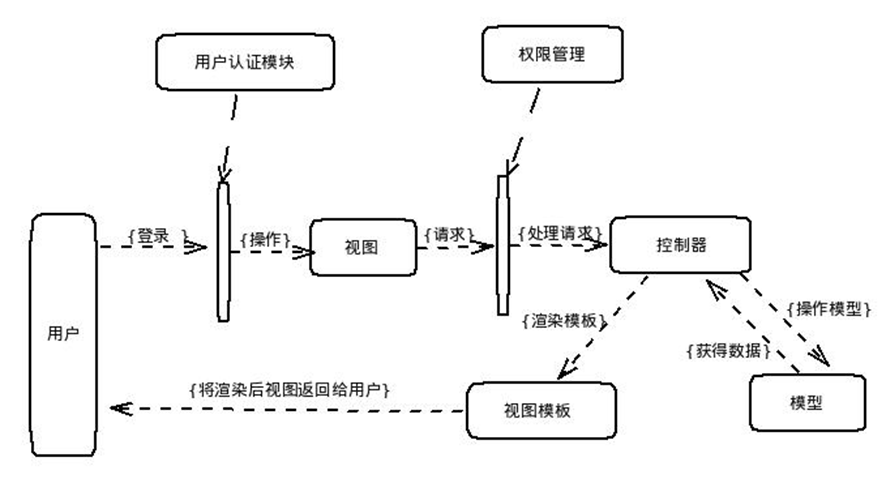
### 前端框架：Bootstrap、JQuery

* 前端CSS框架我们选择Bootstrap，她来自 Twitter，是目前最受欢迎的前端框架。Bootstrap 是基于 HTML、CSS、JAVASCRIPT 的，它简洁灵活，使得 Web 开发更加快捷。它由Twitter的设计师Mark Otto和Jacob Thornton合作开发，是一个CSS/HTML框架。Bootstrap提供了优雅的HTML和CSS规范，它即是由动态CSS语言Less写成。Bootstrap一经推出后颇受欢迎，一直是GitHub上的热门开源项目，包括NASA的MSNBC（微软全国广播公司）的Breaking News都使用了该项目。国内一些移动开发者较为熟悉的框架，如WeX5前端开源框架等，也是基于Bootstrap源码进行性能优化而来。
* 为方便网站前端动态效果的实现，我们选择Javasrcipt框架JQuery，她是继prototype之后又一个优秀的Javascript库。它是轻量级的js库 ，它兼容CSS3，还兼容各种浏览器（IE 6.0+, FF 1.5+, Safari 2.0+, Opera 9.0+），jQuery2.0及后续版本将不再支持IE6/7/8浏览器。jQuery使用户能更方便地处理HTML（标准通用标记语言下的一个应用）、events、实现动画效果，并且方便地为网站提供AJAX交互。jQuery还有一个比较大的优势是，它的文档说明很全，而且各种应用也说得很详细，同时还有许多成熟的插件可供选择。jQuery能够使用户的html页面保持代码和html内容分离，也就是说，不用再在html里面插入一堆js来调用命令了，只需要定义id即可。

## 架构设计

本系统是基于B/S架构使用Python进行开发的，整体的框架可以说就是MVC框架，即分为Model(模型)、View(视图)、Contrller(控制器)三个大的模块，整体框架如下图 ：



系统对用户的认证和认证后的权限进行了管理，用户操作视图页面，向控制器发出请求，加入权限允许，那么控制器处理相应请求，并操作模型获得相关数据，然后用获得的数据去渲染视图的模板，再将渲染后的视图回传给用户。

* 视图

一般将其称作前端，是系统与用户直接交互的环境，提供用户请求服务器的接口，视图模块提供的功能有：用户登录、用户注册、查看公益活动、申请公益活动、创建公益活动、更改个人信息、审查申请等。

* 控制器

可以说是整个系统的核心部分，即相当于大脑功能，它根据前端用户发来的请求，处理相应数据，并做出反馈，是一个中心枢纽。它有用户登录、用户注册、处理申请、创建公益活动等功能模块。

* 模型

负责操作数据库的模块，这里分为两个子模块： data access layer(dal) 和web data layer(wdl)。其中dal定义了对数据库各个表的增、删、改、查这些基本操作，而wdl则是直接定义了每个数据类的数据操作方法，而这些方法是通过调用dal实现的。

## 模块划分

Django是一个基于MVC构造的框架。但是在Django中，控制器接受用户输入的部分由框架自行处理，所以 Django 里更关注的是模型（Model）、模板(Template)和视图（Views），称为 MTV模式。所以，本项目大体上也分为这三大模块。

### Model

模型，也就是数据存取层。在本项目中对应的文件为/philosopher/models.py，分为三个类UserProfile(用户信息)，Activity(活动)，Application(申请表)。

另外，还使用Django框架自带的类User来做为本项目的用户类，优点有二：

* + 符合开发规范，提高了代码的可读性
  + 使用现成框架，节省开发时间

但是，考虑到User中的相关属性相对较少，直接修改底层代码又是非常冒险也不提倡的做法，所以就使用了UserProfile类来封装User并提供其他相关的属性。这样，当需要对用户添加属性时，只需要直接在UserProfile中添加一列就可以了，可扩展性较高。

### View

视图，即业务逻辑层。存取模型及调取恰当模板的相关逻辑。模型与模板之间的桥梁。Django1.4框架中，view层中全部逻辑都是在一个文件views.py中实现的，但是，考虑到本项目中逻辑交互较多，全部存放在一个文件中将会降低代码的可读性以及可维护性，故本项目将View层细分为四大模块，结构如下：

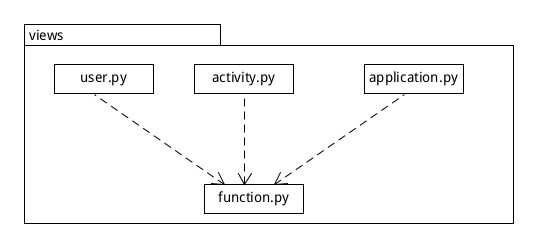
也就是将views文件拆分为一个文件夹，其中

**funcion.py**封装一些基本操作；

**user.py**封装与用户操作相关的操作，如登录，查看个人资料等，同时它也调用function.py来使用其基本函数；

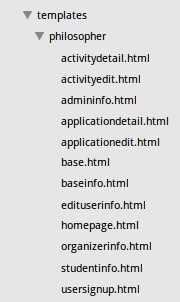
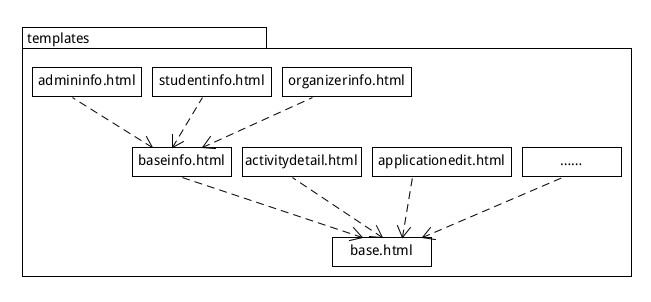
其他的文件也封装各自对应的类的相关操作

其调用关系图如下：

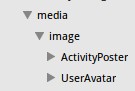


1. Template

模板，即表现层。对应于本项目的/templates/philosopher文件夹，其使用的是Django提供的模板语言，接受后端传递的数据并渲染为html页面。本项目中该层的结构如下左图所示；可以理解为它是页面驱动的，基于每个页面提供一个相应的模板页，同时，他们之间的关系如下右所示



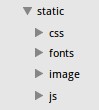
1. **其他相关小模块**

依据Django开发框架的设定，其他模块也都做了较好的分离。

* 表单模块

存放表单的相关信息，在本项目中对应为/philosopher/forms.py。

* 多媒体模块

存放用户上传的图片资料，在本项目中对应的结构有右所示

* 静态资源模块

存放如css，jquery，font等等与前端页面显示相关的资源，其结构如右所示。这样做，一方面可以将前后端合理的分离，方便开发人员的开发；另一方面，将前端相关的代码集中到一个模块，也提高了代码的可读性。

## 技术实现

### Structured programming

* SP简介

结构化程式设计（英语：Structured programming），一种编程典范。它采用子程序、程式码区块（英语：block structures）、for循环以及while循环等结构，来取代传统的 goto。希望借此来改善计算机程序的明晰性、品质以及开发时间，并且避免写出面条式代码。

结构化程序设计提出的原则可以归纳为32个字：自顶向下，逐步细化；清晰第一，效率第二；书写规范，缩进格式；基本结构，组合而成。

* SP实现说明

在本项目中，我们将频繁使用到的基本操作分离为子程序，并将其存放于function.py文件中，需要使用时直接调用函数即可。这样，使得代码清晰，也方便维护。

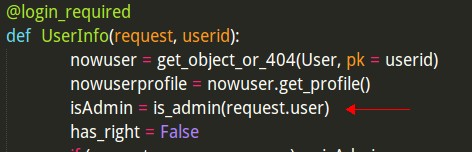
比如，项目中的几乎每一个页面都会用到验证用户身份这样一个功能，这样，项目中便将用户身份验证匹配的操作分离为子程序，如下：



然后，在其他页面中将其import进来，如下



最后，当需要匹配用户身份时，直接调用函数即可，如下

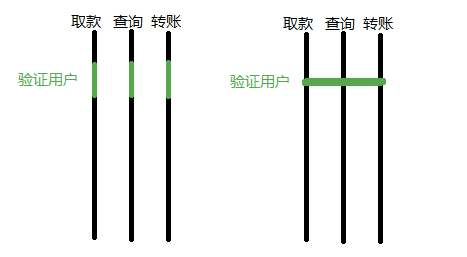


### Aspect-Oriented Programming

* AOP简介

AOP， 即面向切面编程，就是把多个流程中重复出现的业务抽取出来，封装好。例如有两个流程，都需要使用到验证功能，此时我们有两种选择，一种是单独的开发两个流程。另一种方法是先把验证功能开发出来，然后两个流程在到达验证业务的时候就接入到已经开发的验证模块。而这个把公用的模块单独出来，不需要重复开发同样的功能的开发思想就是AOP思想。以下这一幅截取出来的图片就能够很好的说明问题。

个人认为AOP与模块化其实有点相像，但又不完全类似。后者是把每一个功能块独立出来，要使用就调入这个接口。而前者是更抽象的一个概念，像Model\view\Control层一样，AOP就是对特定重用功能再分一层，而一层中可能需要多个模块才能完成这一层的任务



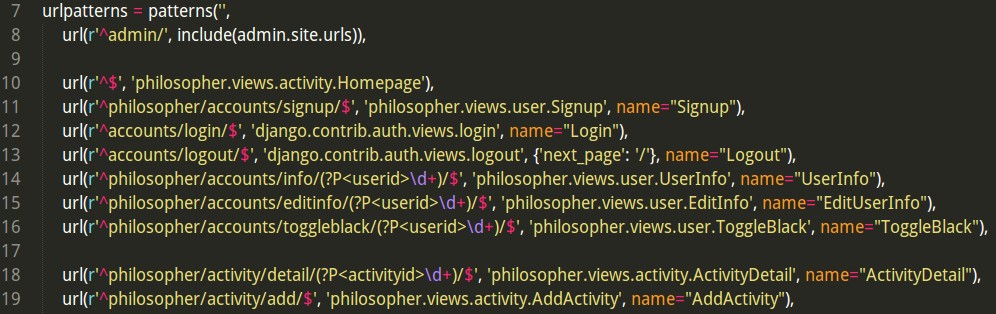
从小而言，可以是重用代码的一种方式。而放大来说，AOP就是一种软件的分层思想，特别用于多人协作的工程。

* AOP实现说明

而在我们项目的开发过程中，由于需求和设计做得比较详尽，在开发前我们就确定了有几个模块是会被几个Use case多次用到的，于是基于AOP的思想，我们就先开发了这些单独的模块，然后重用这些模块代码。

(1)Front Router的配置，对URL识别分发到控制器

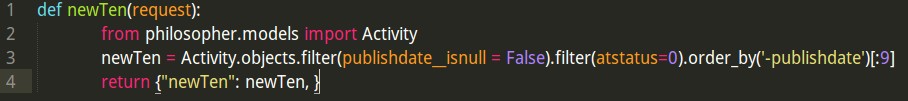
个人认为URL的识别分发可以算是一个AOP的实现，因为对于url请求和控制器调用，如果按照传统的模式，服务器会根据url到对应的路径请求文件，然后再从文件调用控制器，最后完成Conrtrol层的工作。而URL的分发就算是切片编程，在Control层与View层间再切出一层Front Router，而这一层的工作就是对URL进行识别以及分发到控制器，省却流程中的url与控制器识别的步骤。以下是本项目ActivitityManager/urls.py文件中关于url的配置。



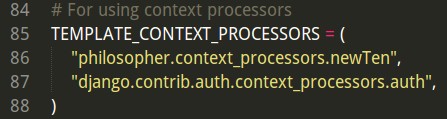
(2)数据请求的拦截器，用于整体数据的添加

而数据请求的拦截器则是基于MVC模式的，由于django使用的是View层是使用template模板的，只有得到数据的导入，模板才能正常工作。而这一个数据请求拦截器则是为了对付一些每个页面都会使用到的数据，如项目中的右侧栏的排行数据资料。这一层在View层与数据请求间划分出来一层，用于添加整体的数据。使我们可以省却每一次取数据都要取一次通用数据的苦况。

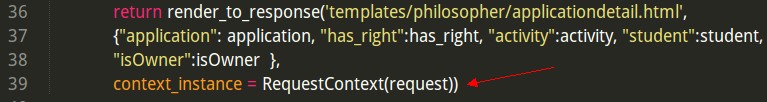
实现的具体方法就是在philosopher应用中加入一个context\_processer.py文件用于存放数据的操作函数。



然后，在settings.py文件中指明该文件的位置



最后，在调用模板渲染时，加入RequestContext



这样，当每次将数据发送到template中时，调用上下文处理器时，便会自动加入newTen的数据，实现每个页面都可以显示前十排行榜

### Object-oriented programming

* OOP介绍

面向对象程序设计（OOP）是一种[程序设计范型](http://baike.baidu.com/view/15027394.htm)，同时也是一种程序开发的方法。具有相同数据和操作的对象可以归纳成类，对象是类的实例。OOP将对象作为程序的基本单元，将程序和数据封装其中，以提高软件的重用性、灵活性和扩展性。

OOP被理解为一种将程序分解为封装数据及相关操作的模块而进行的编程方式。有别于其它编程方式，OOP中的与某数据类型相关的一系列操作都被有机地封装到该数据类型当中，而非散放于其外，因而OOP中的数据类型不仅有着状态，还有着相关的行为。一般OOP都具有以下几个基本理论：

类：定义了一件事物的抽象特点。

封装：用抽象的数据类型将数据和基于数据的操作封装在一起，数据被保护在抽象的数据类型内

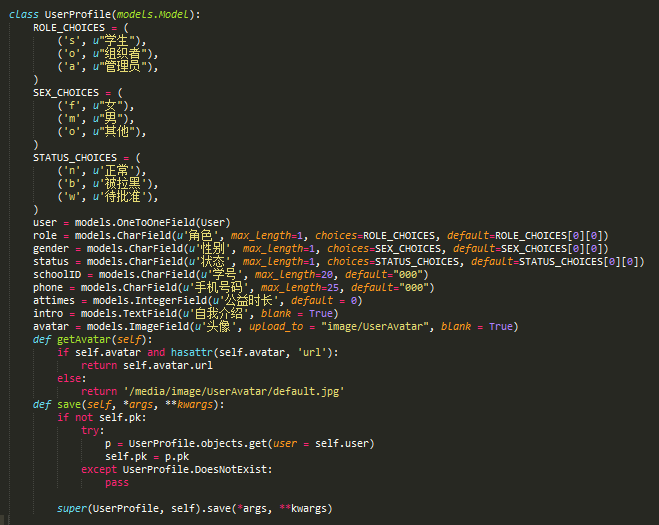
继承：子类拥有父类的所有数据和操作。

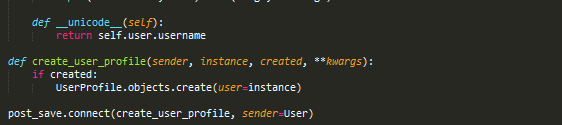
多态：一个程序中同名的不同方法共存的情况。有两种形式的多态：重载和重写。

* OOP实现说明

公益活动管理系统的编程设计采用了python的web框架django，django是由python编写而成的web框架，因此它具有面向对象的所有特点。公益活动管理系统采用面向对象编程技术，提高了软件的重用性、灵活性和扩展性。在设计中我们抽象出了现实生活公益活动管理过程中需要用到的类，将申请表，公益活动，用户等对象抽象为类，简化了复杂的问题，如下：

用户

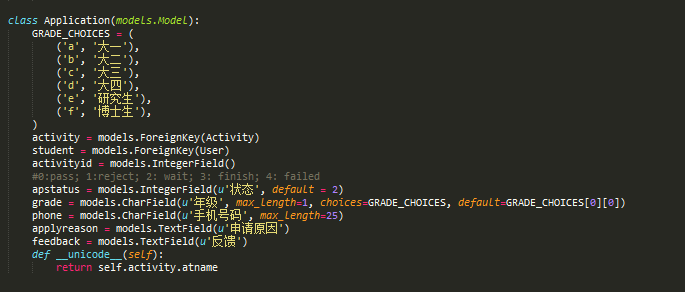




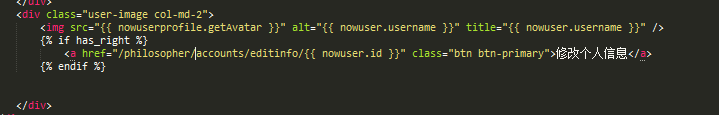
公益活动



申请表

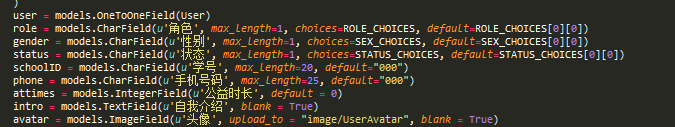


另一方面，公益活动管理系统将类中的数据进行封装。封装是通过限制只有特定类的对象可以访问这一特定类的成员，而它们通常利用接口实现消息的传入传出。这样可以保证数据的安全性，如下：



前端页面只能通过getAvatar获取用户头像。

在公益活动管理系统中，我们还运用了继承的思想，提高了代码的可重用性，在用户类中，我们继承了django自带的user类，并定义了系统中用户应该存在的属性。如：



models.OneToOneField()函数实现了对user的继承。

### MVC

* MVC简介

MVC全名是Model View Controller，是模型(model)－视图(view)－控制器(controller)的缩写，一种软件设计典范，用一种业务逻辑、数据、界面显示分离的方法组织代码，将业务逻辑聚集到一个部件里面，在改进和个性化定制界面及用户交互的同时，不需要重新编写业务逻辑。MVC 分层有助于管理复杂的应用程序，因为您可以在一个时间内专门关注一个方面。MVC主要分为：

Model（模型）：是应用程序中用于处理应用程序数据逻辑的部分。常模型对象负责在数据库中存取数据。

View（视图）：是应用程序中处理数据显示的部分。通常视图是依据模型数据创建的。

Controller（控制器）：是应用程序中处理用户交互的部分。通常控制器负责从视图读取数据，控制用户输入，并向模型发送数据。

* MVC实现说明

公益活动管理系统的编程设计采用了python的web框架django，Django是一个基于[MVC](http://baike.baidu.com/view/31.htm)构造的框架。但是在Django中，控制器接受用户输入的部分由框架自行处理，所以 Django 里更关注的是模型、模板和视图。在该系统中

* models.py 文件主要用Python 类来描述数据表。称为模型。在该文件中定义了用户，活动，申请表三个类。运用这些类，可以进行创建、检索、更新、删除 数据库中的记录而无需写一条又一条的SQL语句。
* views.py文件包含了页面的业务逻辑。
* urls.py 指出了什么样的 URL 调用什么的视图。

template 包含了 html 模板，它描述了页面是如何设计的。

## 网站测试

### 功能测试

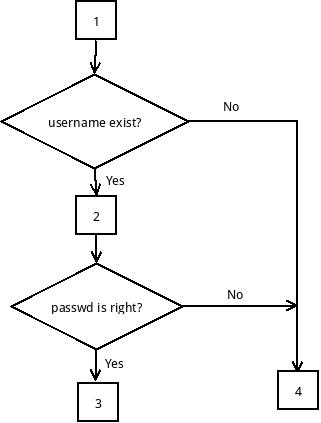
功能测试

由于一个Web程序由多种逻辑层组成——从Http层的request 执行，到表单的确定和处理，再到模板的渲染，所以直接测试一个网站将会是一个比较复杂工作。但是， 通过利用Django测试框架以及齐全的配套工具，我们可以比较容易的模拟request，插入测试数据，检查程序输出和一般情况下的测试。

在Django的测试框架中，测试有两种方式：Doctests 和Unit Tests。本项目采用的是第二种方式，其具体的代码位于/philosopher/tests.py文件中。以下谨描述本网站的用户登录以及用户注册两个用例的测试流程。

1. **用户登录**

**该用例的控制流图如下所示**

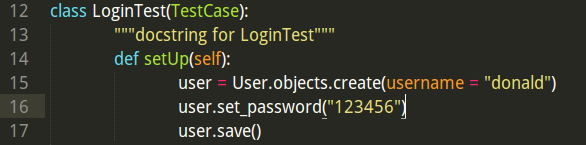


**测试用例：**

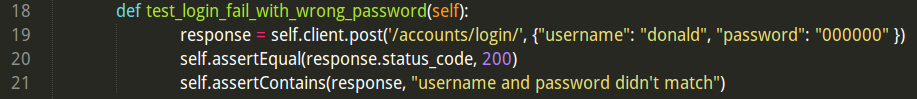
* 用户名不存在
* 密码错误
* 正常登录

**测试函数**

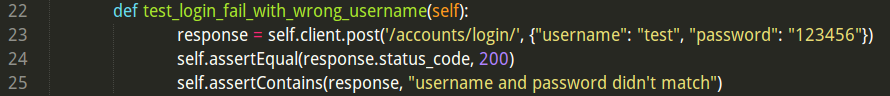
基本环境设置，创建一个用户



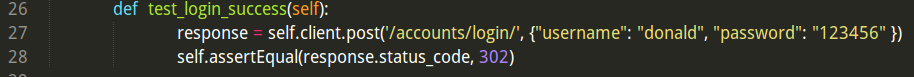
* 用户名不存在



* 密码错误



* 正常登录，由于登录成功会跳转到首页，所以返回码是302

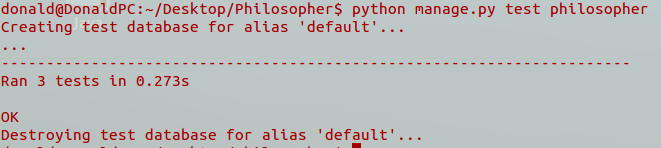


**用例覆盖路径块分别为：**

* {1, 4}
* {1, 2, 4}
* {1, 2, 3}

块覆盖率为：4/(4-0) \* 100% = 100%

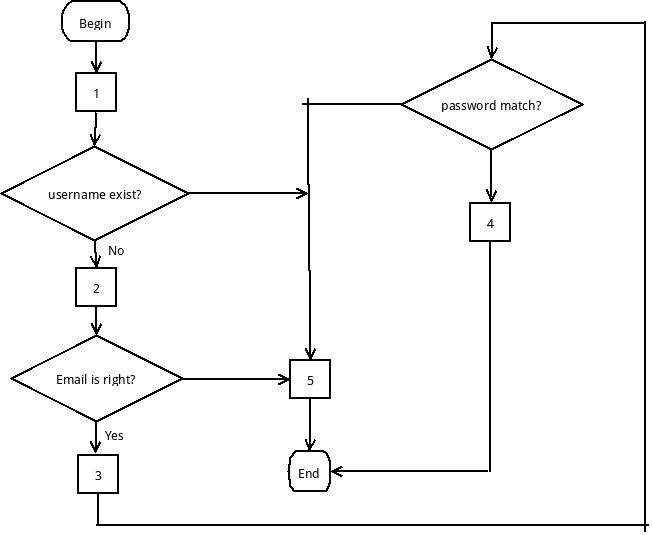
**测试结果**



测试结果符合预期。

1. **用户注册**

**该用例的控制流图如下所示**

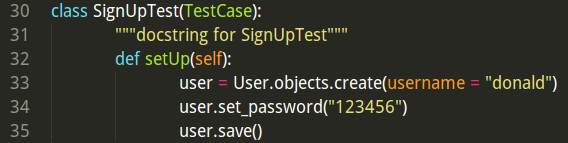


**测试用例：**

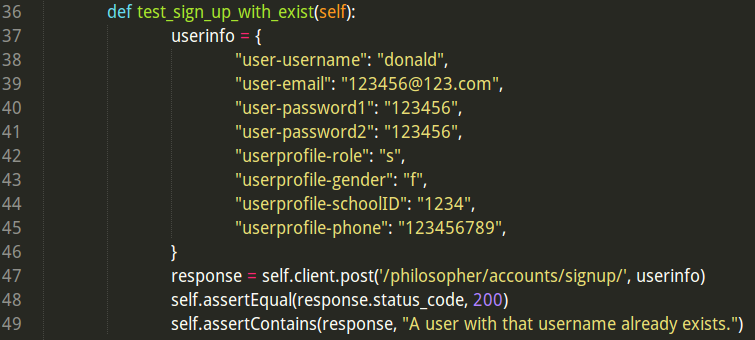
* 用户名已存在
* 邮箱格式错误
* 两次输入密码不一致
* 正常注册

**测试函数**：

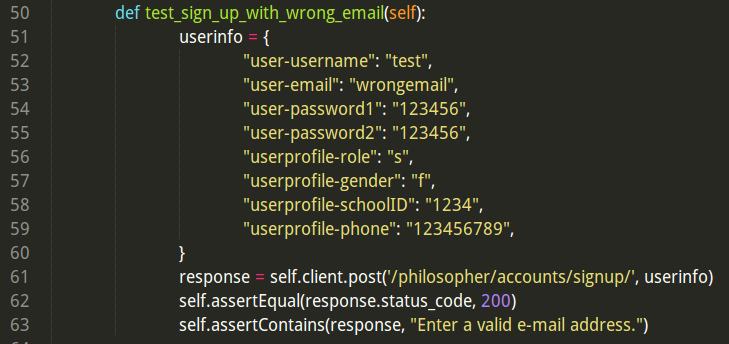
基本环境设置，创建一个用户



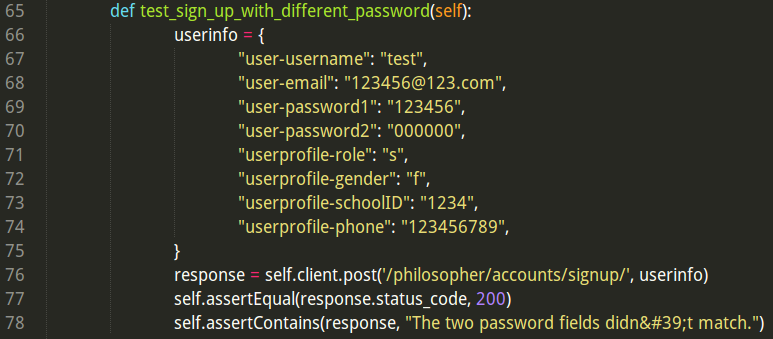
* 用户名已存在



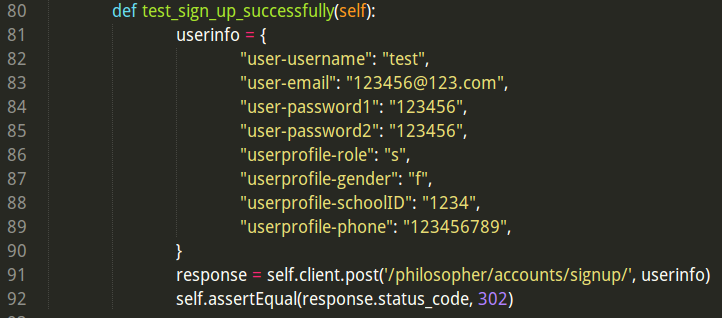
* 邮箱格式错误



* 两次输入密码不一致



* 正常注册，由于注册成功后会自动登录并跳转到首页，所以返回码是302

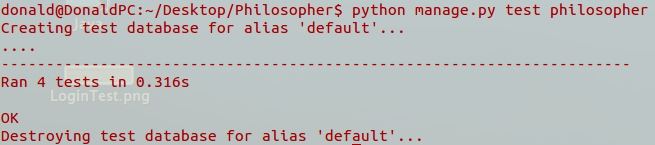


用例覆盖路径块分别为：

* {1, 5}
* {1, 2, 5}
* {1, 2, 3, 4}
* {1, 2, 3, 5}

块覆盖率为：5/(5-0) \* 100% = 100%

**测试结果**

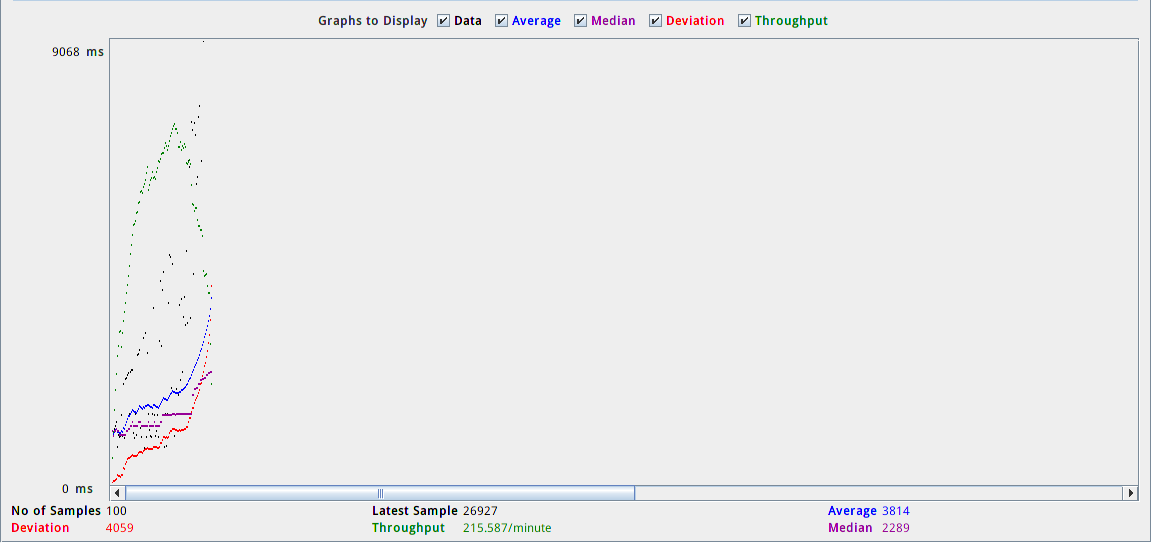


测试结果符合预期。

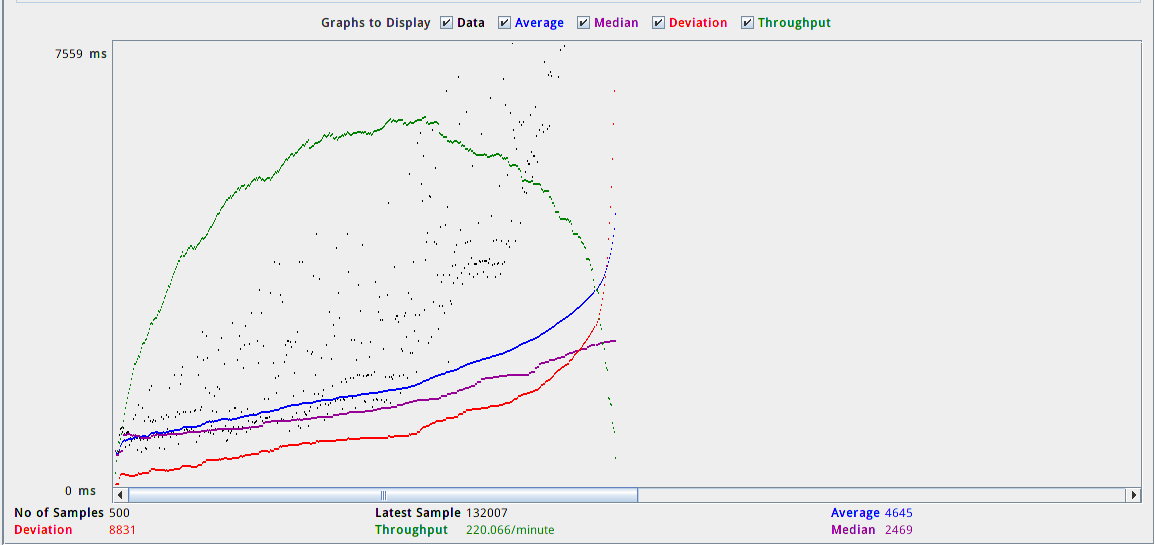
### 负载和压力测试

本项目采用JMeter来对其进行负载/压力测试，创建的线程组的数目依次是100，500，1000，1500，2000，2500，Ramp-Up period为5秒，每个测试计划循环一次。测试计划存储在项目/JmeterTest/路径下，测试的结果及分析如下

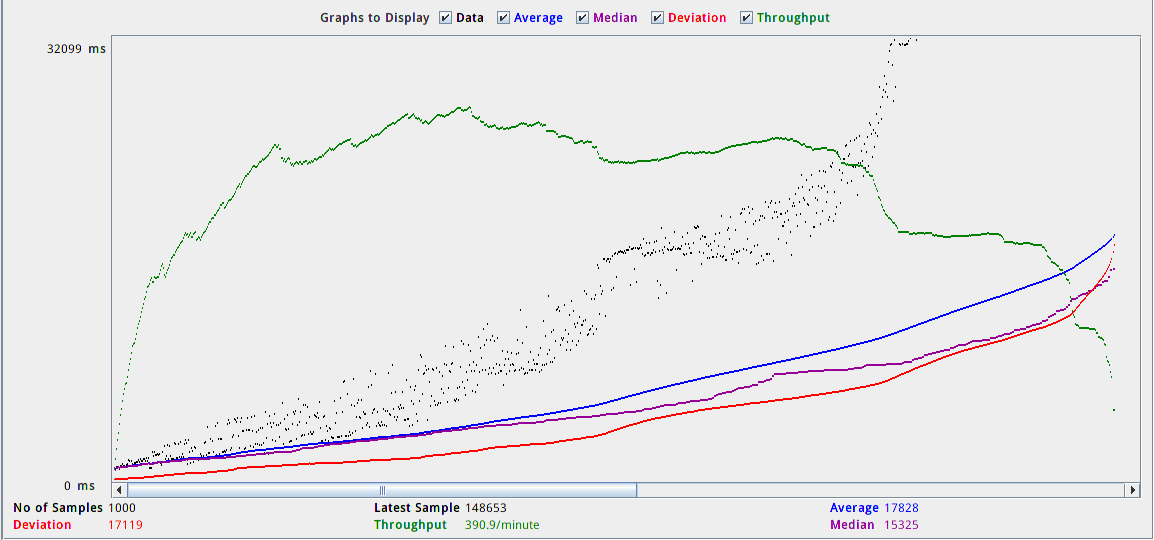
* 线程组为100的结果图



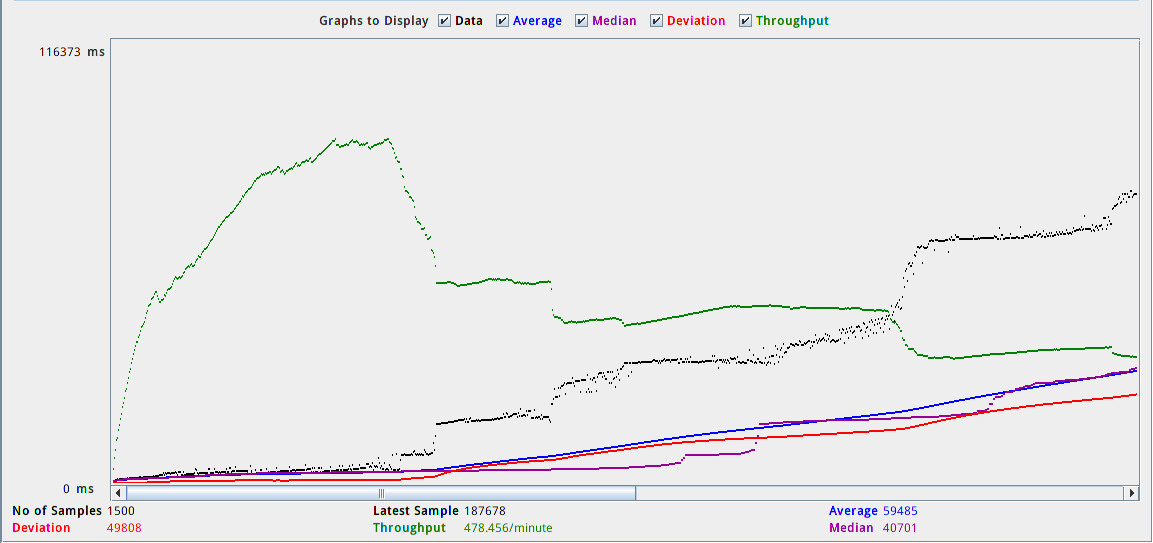
* 线程组为500的结果图



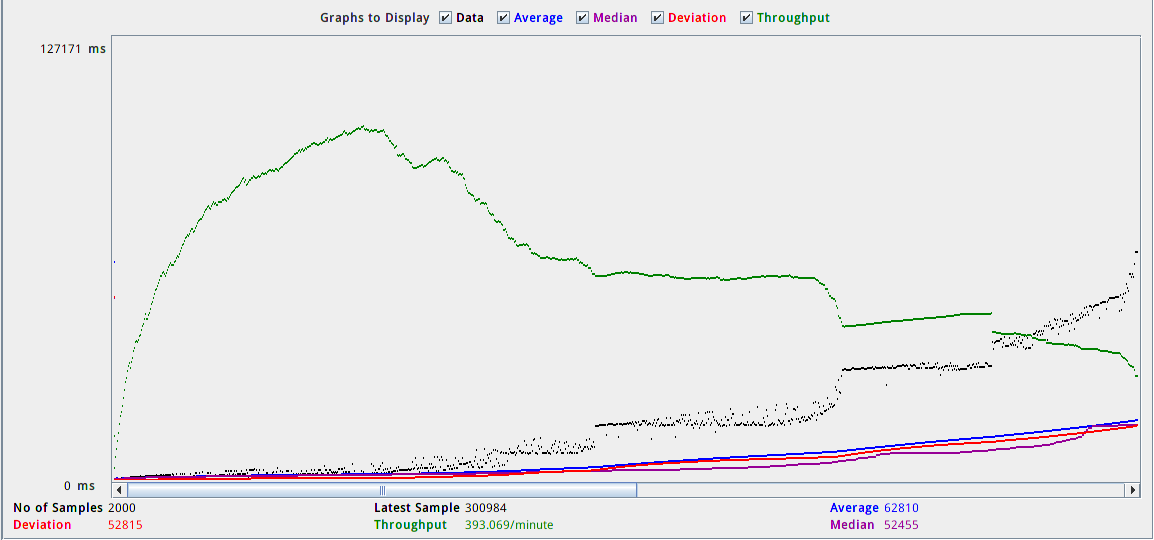
* 线程组为1000的结果图



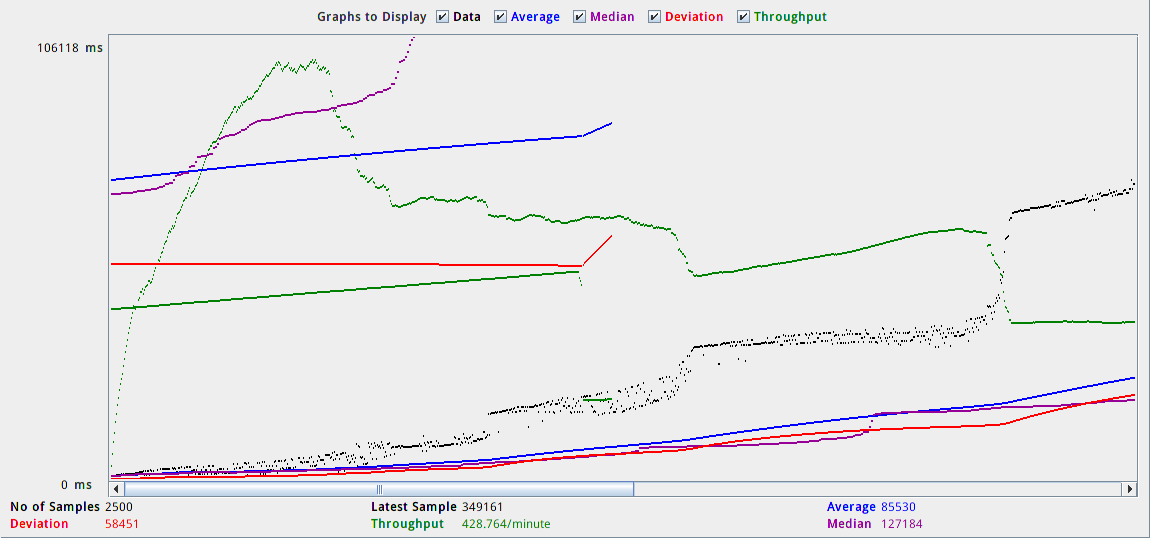
* 线程组为1500的结果图



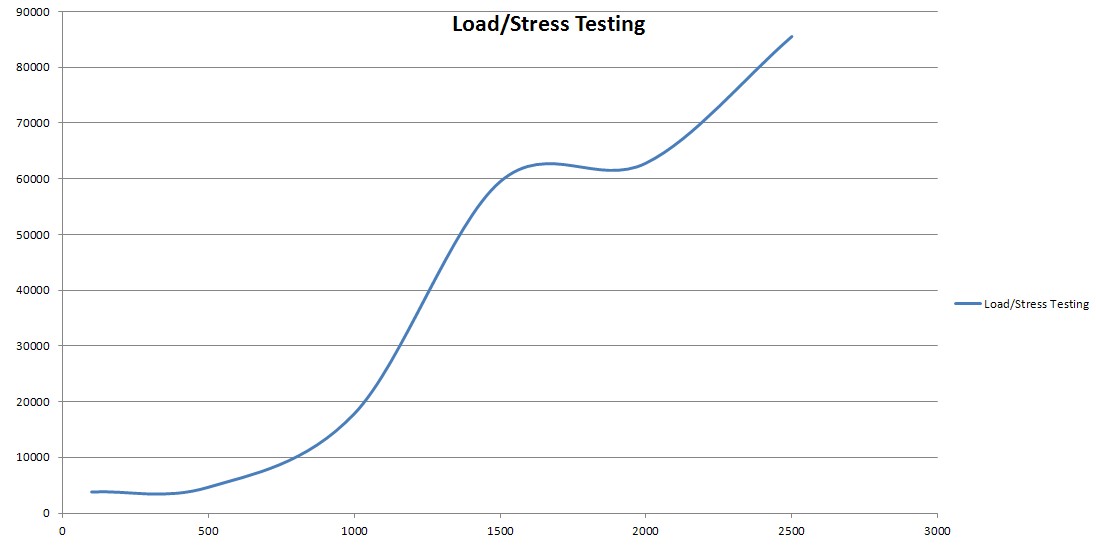
* 线程组为2000的结果图



* 线程组为2500的结果图



* X(线程数)-Y(平均访问时间/ms)的平滑线状图



分析：

从图形结果可知，吞吐量和平均时间一开始基本都是呈上升状态，后来又略有下降。而从整体的线状图可以看出，访问的平均时间随着线程总数的增加而增加。当线程数达到2500时，平均时间增幅较大，其原因在于后面有近100个线程长期属于卡顿状态。可见，这时应该是达到了网站的最高的访问频率，所以网站同一时间能够承受的最高的访问量应该是2500/5=500（Ramp-Up period为5秒），基本满足预期目标。