子夜吴歌

微信公共平台设计文档

子夜吴歌小组

2013

目录

[简介 3](#_Toc376019610)

[总体架构 3](#_Toc376019611)

[结构图 4](#_Toc376019612)

[应用服务器 4](#_Toc376019613)

[功能简介： 4](#_Toc376019614)

[应用服务器架构： 5](#_Toc376019615)

[配置： 5](#_Toc376019616)

[应用： 5](#_Toc376019617)

[通信： 6](#_Toc376019618)

[第三方类库： 8](#_Toc376019619)

[技术实现： 8](#_Toc376019620)

[指数计算： 8](#_Toc376019621)

[微信高级接口实现： 10](#_Toc376019622)

[数据服务器 11](#_Toc376019623)

[功能简介： 11](#_Toc376019624)

[服务器架构 12](#_Toc376019625)

[类及接口说明 12](#_Toc376019626)

[结构 14](#_Toc376019627)

[数据库关系图 14](#_Toc376019628)

[字段说明 14](#_Toc376019629)

[扒取器设计 19](#_Toc376019630)

[交互设计 22](#_Toc376019631)

[账号绑定 22](#_Toc376019632)

[趣味应用 23](#_Toc376019633)

[我的格子 23](#_Toc376019634)

[查询信息 24](#_Toc376019635)

# 简介

我们的微信公共平台叫做”维修助手“（这是由于申请服务号的缘由所致），我们为大家提供了基于学校网络学堂的一系列的公共服务，目前服务器部署在刘强老师的实验室里（166.111.80.7）

# 总体架构

我们的架构分为四部分，分别为微信服务器，应用服务器，数据服务器以及扒取器。

|  |  |
| --- | --- |
| 微信服务器 | 此服务器是微信端的服务器，我们无法操作 |
| 应用服务器 | 通过数据层的数据来提供app服务，返回给微信端 |
| 数据服务器 | 负责数据的存取以及与更后端扒取器的连接 |
| 扒取器 | 负责数据的扒取 |

## 结构图

# 应用服务器

## 功能简介：

CentralServer端即应用服务器，主要功能就是为微信端提供公共平台的应用服务，当微信端发送请求时，应用服务器首先从数据服务器获取数据，然后根据微信端的请求类型组织成服务应用返回给微信用户。

## 应用服务器架构：

### 配置：

应用配置：应用服务器的应用配置在projectManager模块中，这个模块主要负责的是整个微信公共平台应用的全局配置，比如xml模板，appId，appSecret，appToken等。

服务器配置：服务器配置在centralServer模块中，这部分就是Django工程的url，settings等的配置信息。

### 应用：

查询服务：查询服务为用户提供与信息门户以及网络学堂相关的查询服务，包括

用户设置：用户设置包括绑定用户、解除绑定、关注课程等功能

App:包括我的格子、趣味应用等根据个人信息所提供的个性化服务

### 通信：

通信模块：通信模块在requestHelper模块中，主要负责主动请求其他服务器，比如数据服务器和微信服务器。

具体模块功能：

我们程序按模块共分为如下几个模块：xmlHelper，weChatPlatform，exponents，keyHandler，projectManager，requestHelper，static，templates，其余如PIL，poster，qrcode是我们引用的第三方类库。

|  |  |
| --- | --- |
| centralServer | Django工程的配置信息 |
| xmlHelper | 此模块负责解析xml字符串以及通过模板来生成相应的xml字符串 |
| weChatPlatform | 此模块是公共平台的核心模块，主要负责处理各种事件，如eventHandler， textHandler，而handler则是对各种事件的调配处理。此外此模块中还包含了对于事件处理的更具体的辅助操作，比如errorHelper负责错误处理，validationHelper负责对用户的验证，views和viewsIndex处理对html页面的处理 |
| Exponents | 此模块负责各种指数的计算，具体见技术实现段落 |
| projectManager | 此模块负责管理全局参数，包括xml模板，appId，appSecret，appToken等 |
| requestHelper | 此模块负责与其他服务器的通信，baseReqHelper和reqHelper这部分负责模拟登陆微信公共平台，从而实现高级借口的伪实现，forwardHelper封装了放松request请求的相关函数，tokenHelper负责更新微信的appToken的管理操作，messageHelper是基于baseReqHelper和reqHelper封装了微信主动推送的高级借口的功能实现 |
| static | 此模块放置了静态文件 |
| templates | 此模块放置了html模板 |

### 第三方类库：

|  |  |
| --- | --- |
| PIL | 图像处理的Python库 |
| poster | 用于模拟http请求的Python库 |
| qrcode | 二维码操作的Python库 |

## 技术实现：

### 指数计算：

**屌丝指数：**

基本思路是根据用户的皮肤颜色来推测其屌丝程度，皮肤偏黑则屌丝指数偏高，反之自然是高富帅与白富美了= =

扒取得到用户的头像图片，进行图片增强之后，将图片转换成256级灰度图，计算图片像素的平均灰度。

因为用户图片的背景相同，故此处对图片背景不做特殊处理。

由此每个用户都得到了一个对应的处理图片后的数值，统计这些数值，得到其大致分布，确定数值所处区间平均分为5份，即为5个等级。

**学霸指数：**

根据用户所选课程的数量以及学分数量进行线性组合，统计所有用户的数值作为其学霸指数的评判标准，对于整体分布区间划分为5级。

**明星脸：**

使用google的图片搜索算法。使用“感知哈希”算法，为图片生成一个指纹(字符串格式), 两张图片的指纹越相似, 说明两张图片就越相似。扒取数十张老师的头像图片作为参考图库。

主要的步骤如下：

1、将图片缩小到8x8的尺寸, 总共64个像素. 这一步的作用是去除各种图片尺寸和图片比例的差异, 只保留结构、明暗等基本信息。

2、将缩小后的图片，转为64级灰度。也就是说，所有像素点总共只有64种颜色。

3、计算图片像素点的平均值，将每个像素的灰度，与平均值进行比较。大于或等于平均值，记为1；小于平均值，记为0。这个时候生成的像素序列即为该图的指纹。

4、得到指纹以后，就可以对比不同的图片，看看64位中有多少位是不一样的。如果两张图不同的位数越少则说明两张图越相似。

### 微信高级接口实现：

由于我们微信公共平台并非认证的服务号，所以高级接口无法使用，但是这些高级接口某些事很有实用价值，所以我们通过模拟登陆的方法来实现这些高级借口，主要思路就是通过模拟公共平台的网页之后，模拟post请求来实现这些功能，比如主动推送。

但是有一个问题就是，这里的模拟post请求都是需要用户的fakeId的，而微信服务器只会告诉我们经过加密的openId，所以我们最大的问题就是获取fakeId。为此我们设计了验证码，通过让用户发送验证码来判断用户的身份，从而在网页上扒取fakeId。

拓展app介绍：

我的格子：此app服务可以看做我们平台所提供的核心服务，因为我们主要是基于网络学堂的服务，但是如果提供的服务和网络学堂太过相似，那么我们的平台效果不会太好，所以我们主力推出了此功能。我们的主要思路在于，用户对于网络学堂最为关心的项目在于作业服务，所以我们此功能中可以清晰地查询自己的deadline，这样的服务既是用户十分需求的，也是网络学堂的盲点。

趣味应用：此部分是为了增添平台的趣味性所推出的一系列小应用，在之前的部分中已经介绍过。

个人名片：这同样是一个很实用的小应用，通过我们从网络学堂上取得的个人信息，我们为用户自动生成了二维码名片，这样其他微型用户只需轻轻一扫，即可将你的信息存入手机，方便实用。

# 数据服务器

## 功能简介：

DataServer端为数据服务器，主要功能就是爬取用户的相关数据并存储在数据库中，在收到前端服务器的请求后返回相关数据。

## 服务器架构

信息爬取：模拟html请求从网络上获得返回的html代码，解析后获得相关的信息

信息存储：对于从网络上爬取的信息进行处理后按层次存储在mongodb数据库中

信息查询：提供各种查询接口，可以返回各种前端服务器所需要的数据

## 类及接口说明

grubber包：

|  |  |  |
| --- | --- | --- |
| 父类 | 子类 | 说明 |
| hunter | hunter\_learn | 爬取网络学堂上的各类信息 |
|  | hunter\_academic | 爬取信息门户上的各类信息 |

mythread包：

|  |  |  |
| --- | --- | --- |
| 父类 | 子类 | 说明 |
| thread | thread\_grubber | 用户绑定账户时的信息获取线程 |
|  | thread\_update | 用户信息更新时的信息获取线程 |

store包：

|  |  |  |  |
| --- | --- | --- | --- |
| 父类 | 子类 | 接口 | 说明 |
| store | store\_learn | learn\_store | 网络学堂信息爬取及储存 |
| course\_attention\_set | 设置用户关注课程 |
| store\_academic | academic\_store | 信息门户信息爬取及储存 |
| store\_wechat | fake\_id\_store | 用户fake\_id储存 |

query包：

|  |  |  |  |
| --- | --- | --- | --- |
| 父类 | 子类 | 接口 | 说明 |
| query | query\_learn | course\_list\_query | 返回课程列表 |
| course\_info\_all\_query | 返回课程信息列表 |
| homework\_info\_query | 返回课程信息列表 |
| notice\_info\_query | 返回公告信息列表 |
| files\_info\_query | 返回文件信息列表 |
| my\_table\_query | 查询近期未交作业及公告，返回一个以日期为key的dict |
| query\_academic | person\_info\_query | 返回个人信息 |
| term\_list\_query | 返回学期列表 |
| score\_info\_query | 返回成绩列表 |
| query\_wechat | fake\_id\_query | 返回用户fake\_id |
| last\_time\_query | 返回用户最近更新时间 |
| update\_info\_query | 返回用户登录信息 |

## 结构

### 数据库关系图

### 字段说明

User

|  |  |  |
| --- | --- | --- |
| 字段 | 类型 | 说明 |
| user\_name | string | 用户名 |
| use\_password | string | 密码 |
| user\_id | string | 微信id（用户辨识） |
| fake\_id | string | 微信id（主动推送） |
| last\_time | float | 最近更新信息时间 |
| person\_info | PersonInfo | 个人基本信息 |
| course\_attention | list<string> | 关注课程列表 |
| course\_info | list< Course > | 课程信息列表 |
| learn\_info | list<Special> | 课程内容列表 |
| timetable | list< SingleCourse > | 个人课程表 |
| course\_score | list< CourseScoreInfo > | 成绩列表 |

PersonInfo

|  |  |  |
| --- | --- | --- |
| 字段 | 类型 | 说明 |
| real\_name | string | 姓名 |
| sex | string | 性别 |
| major | string | 专业 |
| student\_number | string | 学号 |
| phone | string | 手机 |
| nation | string | 国籍 |
| teach\_class | string | 教学班 |
| department | string | 院系 |
| birth\_date | string | 出生日期 |
| email | string | 邮箱 |
| identification | string | 身份证号 |
| charge\_type | string | 收费类型 |
| is\_register | string | 是否注册 |
| is\_minor | string | 是否辅修 |
| is\_second\_degree | string | 是否第二学位 |

Course

|  |  |  |
| --- | --- | --- |
| 字段 | 类型 | 说明 |
| caption | string | 课程名称 |
| id | string | 课程id |
| file\_unread | string | 未读文件数量 |
| homework | string | 未交作业数量 |
| notice\_unread | string | 未读公告数量 |

Special

|  |  |  |
| --- | --- | --- |
| 字段 | 类型 | 说明 |
| caption | string | 课程标题 |
| notice | list< SpecialNotice> | 公告信息 |
| files | list< SpecialFiles> | 文件信息 |
| homework | list< SpecialHomework> | 作业信息 |
| resources | list< SpecialResources> | 资源信息 |

SpecialNotice

|  |  |  |
| --- | --- | --- |
| 字段 | 类型 | 说明 |
| link | string | 公告链接 |
| caption | string | 公告标题 |
| teacher | string | 公告教师 |
| date | string | 公告日期 |
| text | string | 公告内容 |

SpecialFiles

|  |  |  |
| --- | --- | --- |
| 字段 | 类型 | 说明 |
| link | string | 文件链接 |
| caption | string | 文件标题 |
| note | string | 文件说明 |
| size | string | 文件大小 |
| date | string | 文件上传日期 |

SpecialHomework

|  |  |  |
| --- | --- | --- |
| 字段 | 类型 | 说明 |
| caption | string | 作业标题 |
| state | string | 作业状态 |
| link | string | 作业链接 |
| date | string | 作业发布日期 |
| deadline | string | 作业截止日期 |
| size | string | 作业文件大小 |
| note | string | 作业说明 |
| file | string | 作业文件名称 |
| submit\_file | string | 提交文件名称 |

SpecialResources

|  |  |  |
| --- | --- | --- |
| 字段 | 类型 | 说明 |
| link | string | 资源链接 |
| caption | string | 资源标题 |
| note | string | 资源说明 |

SingleCourse

|  |  |  |
| --- | --- | --- |
| 字段 | 类型 | 说明 |
| caption | string | 课程标题 |
| time | string | 课程时间 |
| duration | string | 课程时长 |
| type | string | 课程类型 |
| day | string | 课程日期 |

CourseScoreInfo

|  |  |  |
| --- | --- | --- |
| 字段 | 类型 | 说明 |
| school\_term | string | 课程学期 |
| course\_caption | string | 课程标题 |
| course\_point | string | 课程学分 |
| score | string | 课程成绩 |

## 扒取器设计

扒取器运行在数据服务器上，作为最底层的数据根源。主要是利用了python脚本模拟登陆了清华大学网络学堂与清华大学信息门户，并扒取了相应的课程、个人、成绩等信息。

扒取器采用面向对象的代码设计，利用类继承的特性简化代码量并方便拓展。

#### 类图设计

#### 具体介绍

##### Hunter类

完成统一操作，提供抽象接口。

|  |  |
| --- | --- |
| def\_\_init\_\_(self,username,password,useBuffer=False) | 构造函数——完成login操作 |
| def \_\_del\_\_(self) | 析构函数——完成logout操作 |
| def open(self, turl) | 获取HTML信息 |
| def getMessageC(self, content, r) | 从content中截取有用的信息 |
| defgetMessage(self,add,r,coding='utf-8',deal=lazyDeal) | 获取信息 |

##### Hunter\_academic类

扒取清华大学信息门户相关内容。

具体方法：通过模拟登陆得到此次登陆的有效token，利用token去访问网页并对返回的网页扒取有效信息。

代码结构：

在类开头编译好扒取器的正则表达式，以加快扒取速度。

|  |  |
| --- | --- |
| def login(self, username, password) | 模拟登陆info |
| def logout(self) | 登出info |
| def getBasicInfo(self) | 获得个人基本信息 |
| def getCourseInfo(self) | 获得个人课表 |
| def getPersonInfo(self, image\_name) | 获得个人详细信息 |
| def getScoreFull(self) | 获得个人成绩 |
| def refreshCache(self) | 模拟刷新 |
| def specialfunc(self, listdata) | 格式化字符串 |

##### Hunter\_learn

扒取网络学堂相关信息，具体方法与hunter\_academic类似。

|  |  |
| --- | --- |
| def login(self, username, password) | 模拟登陆info |
| def logout(self) | 登出info |
| def datadeal(self, li, Host=DEFAULT\_HOST) | 字符串处理 |
| def getInfo(self) | 获取课程总体信息 |
| def getSpecial(self, id, specification) | 获取课程公共、文件、作业、资源 |
| def getHomework(self, link) | 获取作业具体内容 |
| def getNotice(self, link) | 获取公共具体内容 |

##### Debuger.py

因为扒取过程较为复杂，特别是正则表达式的编写极容易出错，因此编写Debuger.py工具方便调试，主要是实现了一个格式化JSON数据显示的函数printer()，可以清楚的打印出目前扒取出来的数据。

# 交互设计

我们在界面的设计上十分用心，致力于设计用户使用舒适、具有观赏性的界面。

## 账号绑定

登录界面的模块大小、间距调整得恰倒好处，符合人的审美习惯和使用习惯。



## 趣味应用

明星脸、学霸指数、屌丝指数页面的配色、配图使人眼前一亮。



## 我的格子

课程格子界面仿佛一张张便利贴，点击未交作业或者未交公告弹出的窗口也是使用方便。



## 查询信息

作业列表、公告列表、文件列表等页面利用mobile jquery的demo实现，经典耐看。

