

Pre:

- (1)python3.x的安装，网上有
- (2)工具的选择，建议sublime text3
- (3)mysql的安装，根据不同环境，网上有

```
(1)安装虚拟环境
pip3 install virtualenv
(2)创建虚拟环境
python3 -m venv ll_env
(3)激活虚拟环境
source ll_env/bin/activate (关闭是deactivate)
```

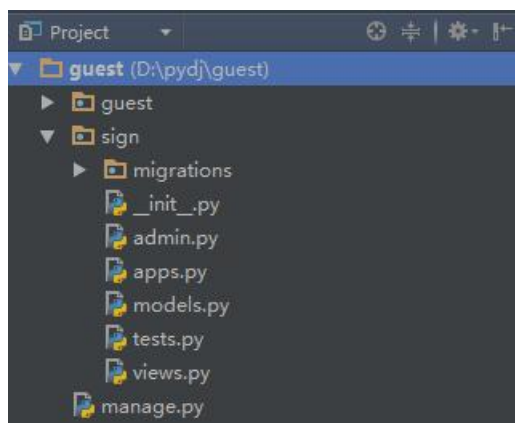
pip3 install Django

```
django-admin startproject guest
```

4.创建应用

```
cd guest
python3 manage.py startapp sign
```

注：django-admin 和 python3 manage.py 的一些命令很相像



5.项目运行

```
cd guest
python3 manage.py runserver
python3 manage.py runserver 127.0.0.1:8001 （使用其它端口的方法）
```

6.将应用添加到项目中

我们首先需要配置一下 guest/settings.py 文件，将 sign 应用添加到项目中。
setting.py 里的 installed_apps

```
settings.py
.....
# Application definition

INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',
    'sign', #添加 sign 应用
]
.....
```

7.设置路由（7、8、9简单的了解一下MVT架构）

打开 guest/urls.py 文件添加该目录。

urls.py

```
.....

from django.conf.urls import url
from django.contrib import admin
from sign import views    #导入 sign 应用 views 文件

urlpatterns = [
    url(r'^admin/', admin.site.urls),
    url(r'^index/$', views.index), #添加 index/路径配置
]
```

8.设置视图

views.py

```
from django.shortcuts import render

# Create your views here.
def index(request):
    return render(request, "index.html")
```

9.使用模版

在应用 sign/目录下创建 templates/index.html 文件。

index.html

```
<html>
  <head>
    <title>Django Page</title>
  </head>
  <body>
    <h1>Hello Django!</h1>
  </body>
</html>
```

10.系统表的设计

进入sign里的models.py

```

from django.db import models

# Create your models here.

#发布会表
class Event(models.Model):
    """docstring for Event"""
    #发布会标题
    name = models.CharField(max_length = 100)
    #参加人数
    limit = models.IntegerField()
    #状态
    status = models.BooleanField()
    #地址
    address = models.CharField(max_length = 200)
    #发布会时间
    start_time = models.DateTimeField('events time')
    #创建时间(自动获取当前时间)
    create_time = models.DateTimeField(auto_now = True)

    #__str__()方法告诉python如何将对象以str的方式显示出来
    def __str__(self):
        return self.name

#嘉宾表

class Guest(models.Model):
    """docstring for Guest"""
    #关联发布会id
    #创建一个外键，通过它可以获取发布会表
    event = models.ForeignKey(Event)
    #姓名
    realname = models.CharField(max_length = 64)
    #手机号
    phone = models.CharField(max_length = 16)
    #邮箱
    email = models.EmailField()
    #签到状态
    sign = models.BooleanField()
    #创建时间(自动获取当前时间)
    create_time = models.DateTimeField(auto_now = True)

class Meta:
    """docstring for Meta"""
    unique_together = ("event", "phone")

```

#用于告诉python如何将对象以str的形式显示出来

```
def __str__(self):  
    return self.realname
```

最后，对于一场发布会来说，一般会选择手机号作为一位嘉宾的验证信息，所以，对于一场发布会来说，手机号必须是唯一。除了嘉宾 id 外，这里通过发布会 id +手机号来做为联合主键。

__str__()方法告诉 Python 如何将对象以 str 的方式显示出来。所以，为每个模型类添加了__str__()方法。(如果读者使用的是 Python2.x 的话，这里需要使用__unicode__())

| | |
|----------------------------|--|
| AutoField | 用于存放 integer 类型的数字。 |
| BooleanField | 用于存放布尔类型的数据 (True 或 False) |
| CharField | 用于存放字符型的数据，需要指定长度 max_length。 |
| CommaSeparatedIntegerField | 用于存放用逗号隔开的 integer 类型的数据。 |
| .DateField | 日期型，必须是“YYYY-MM-DD”格式 |
| DateTimeField | 日期时间型，必须是“YYYY-MM-DD HH:MM[:ss[.uuuuuu]][TZ]”格式。 |
| DecimalField | 小数型，用于存放小数的数字。 |
| EmailField | 电子邮件类型 |
| FilePathField | 文件路径类型，FilePathFields must have either 'allow_files' or 'allow_folders' set to True. |
| FloatField | 浮点型。用于存放浮点型数据。 |
| IntegerField | 用于存放 integer 类型的数字。 |
| BigIntegerField | 用于存放大 integer 类型的数字，最大数支持：9223372036854775807 |
| GenericIPAddressField | 存放 IP 地址的类型，IPv4 和 IPv6 地址，字符串格式。 |
| NullBooleanField | value must be either None, True or False. |
| PositiveIntegerField | Positive integer |
| PositiveSmallIntegerField | Positive small integer |
| SlugField | 需要定义 max_length 值。 |
| SmallIntegerField | Small integer |
| TextField | 用于存放文本类型的数据。 |
| TimeField | 时间类型。“HH:MM[:ss[.uuuuuu]]”格式 |
| URLField | 用于存放 URL 地址 |
| BinaryField | Raw binary data |

11.数据库迁移

```
D:\pydj\guest>python3 manage.py makemigrations sign
```

```
Migrations for 'sign':
```

```
sign/migrations/0001_initial.py:
```

61

虫师原创----<http://fmg.cnblogs.com>

- Create model Event
- Create model Guest
- Alter unique_together for guest (1 constraint(s))

```
D:\pydj\guest>python3 manage.py migrate
```

```
Operations to perform:
```

```
Apply all migrations: admin, auth, contenttypes, sessions, sign
```

```
Running migrations:
```

```
Applying sign.0001_initial... OK
```

12.创建后台超级管理员账号(admin/admin123456)

```
D:\pydj\guest>python3 manage.py createsuperuser
```

```
Username (leave blank to use 'fmgj'): admin #输入登录用户名
```

78

虫师

```
Email address: admin@mail.com #输入用户邮箱
```

```
Password: #输入登录密码
```

```
Password (again): #再次输入用户密码
```

```
Superuser created successfully.
```

13.admin后台管理

打开../sign/admin.py 文件。

admin.py

```
from django.contrib import admin
from sign.models import Event, Guest

# Register your models here.
admin.site.register(Event)
admin.site.register(Guest)
```

这些代码通知 admin 管理工具为这些模块逐一 供界面。

登录 admin 后台:<http://127.0.0.1:8000/admin/> (admin/admin123456)

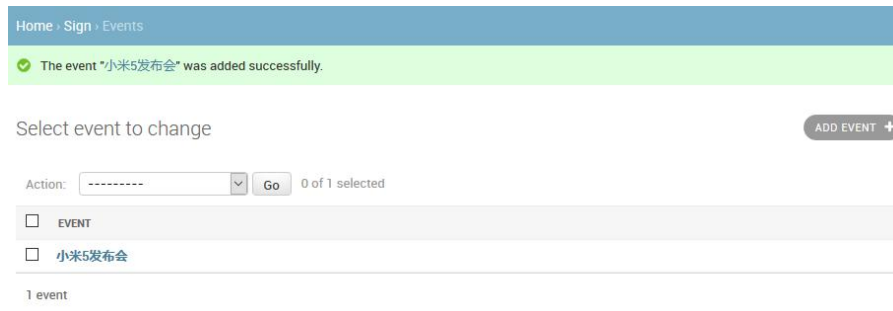
(1)添加发布会

因为：

#__str__()方法告诉python如何将对象以str的方式显示出来

```
def __str__(self):
    return self.name
```

所以：



显示的发布会数据只有发布会名称

(2)如何显示多条信息呢？继续修改

```
# Register your models here.
class EventAdmin(admin.ModelAdmin):
    list_display = ['name', 'status', 'start_time','id']

class GuestAdmin(admin.ModelAdmin):
    list_display = ['realname', 'phone','email','sign','create_time','event']

admin.site.register(Event,EventAdmin)
admin.site.register(Guest,GuestAdmin)
```

用 EventAdmin 选项注册Event 模块
用GuestAdmin选项注册Guest模块

Home > Sign > Events

Select event to change
ADD EVENT +

Action: Go 0 of 1 selected

| <input type="checkbox"/> | NAME | STATUS | EVENTS TIME | ID |
|--------------------------|--------|--------|-----------------------|----|
| <input type="checkbox"/> | 小米5发布会 | ✔ | Dec. 10, 2016, 2 p.m. | 1 |

1 event

Home > Sign > Guests

Select guest to change
ADD GUEST +

Action: Go 0 of 1 selected

| <input type="checkbox"/> | REALNAME | PHONE | EMAIL | SIGN | CREATE TIME | EVENT |
|--------------------------|----------|-------------|---------------|------|--------------------------|--------|
| <input type="checkbox"/> | jack | 13800110011 | jack@mail.com | ✖ | Nov. 4, 2016, 11:24 p.m. | 小米5发布会 |

1 guest

(3)如果我要定制呢，增加搜索栏和过滤器


```
# Register your models here.
class EventAdmin(admin.ModelAdmin):
    list_display = ['name', 'status', 'start_time', 'id']
    search_fields = ['name'] #搜索栏
    list_filter = ['status'] #过滤器
```

64

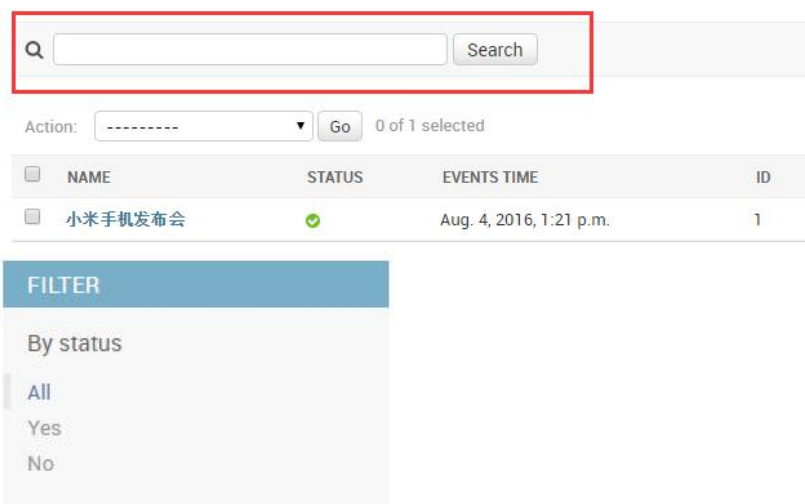
虫师原创----<http://fnng.cnblogs.com>

```
class GuestAdmin(admin.ModelAdmin):
    list_display = ['realname', 'phone', 'email', 'sign', 'create_time', 'event']
    search_fields = ['realname', 'phone'] #搜索栏
    list_filter = ['sign'] #过滤器
.....
```

list_display, 它是一个字段名称的数组, 用于定义要在列表中显示哪些字段。当然, 这些字段名称必须是模型中的 Event()类定义的

以下第一个是搜索栏, 第二个是过滤器

Select event to change



The screenshot shows the Django admin interface for the 'Event' model. At the top, there is a search bar with a magnifying glass icon and a 'Search' button. Below the search bar, there is an 'Action:' dropdown menu with a 'Go' button and a status '0 of 1 selected'. The main table has columns: NAME, STATUS, EVENTS TIME, and ID. There is one row with the name '小米手机发布会', status 'Yes' (indicated by a green checkmark), time 'Aug. 4, 2016, 1:21 p.m.', and ID '1'. On the left side, there is a 'FILTER' sidebar with the title 'By status' and three options: 'All', 'Yes', and 'No'.

14.基础数据访问

python3 manage.py shell

(1)插入数据

```
>>> Guest.objects.create(realname='andy',phone=13611001101,email=
'andy@mail.com',sign=False,event_id=3)
```

(2)查询数据

```
>>> Event.objects.get(name='发布会').address
```

(3)删除数据

```
>>> Guest.objects.get(phone='13611001101').delete()
```

(4)更新数据

```
>>> Guest.objects.select_for_update().filter(phone='13611001101').update(
realname='andy')
```

15.配置MySQL

(1)mysql的安装

Mac只需要解压就可以了，具体过程看自己的文档，或者是百度一下

(2)进入mysql

```
C:\Users\fnngj>mysql -u root -p
Enter password: *****
Welcome to the MySQL monitor.  Commands end with ; or \g.
Your MySQL connection id is 5
```

(3)mysql的基本操作

```
mysql> show databases; #查看当前数据库下面的所有库
```

```
+-----+
| Database          |
+-----+
| information_schema |
| mysql              |
| performance_schema |
| test               |
+-----+
```



```

        'ENGINE': 'django.db.backends.mysql',
        'HOST': '127.0.0.1',
        'PORT': '3306',
        'NAME': 'guest',
        'USER': 'root',
        'PASSWORD': '123456',
        'OPTIONS': {
            'init_command': "SET sql_mode='STRICT_TRANS_TABLES'",
        },
    }
}

```

配置信息从上到下依次是驱动(ENGINE), 主机地址(HOST), 端口号(PORT), 数据库(NAME), 登录用户名(USER), 登录密码(PASSWORD)。

关于, sql_mode 的设置, 请参考 Django 文档。

<https://docs.djangoproject.com/en/1.10/ref/databases/#mysql-sql-mode>

(2)数据库同步

注意:切换了数据库后, 之前 Sqlite3 数据库里的数据并不能复制到 MySQL 中, 所以需要重新进行数据库同步, 使数据模型重新在 MySQL 数据库中生成表。

Pre:先换驱动

但是因为 Django 在连接 MySQL 数据库时默认使用的是 MySQLdb 驱动, 然而我们没有安装该驱动, 因为它并不支持 Python3, 我们现在安装的是 PyMySQL 驱动, 所以在.../guest/__init__.py 目录下添加:

```

import pymysql
pymysql.install_as_MySQLdb()

```

Then: 数据库同步

python3 [manage.py](#) migrate

Last: 重新创建超级管理员

python3 [manage.py](#) createsuperuser

18.mysql管理工具

这里我选择navicat, 请自行下载

19.常见状态码

| | |
|---------------------------|---|
| 200 OK | //客户端请求成功 |
| 400 Bad Request | //客户端请求有语法错误，不能被服务器所理解 |
| 401 Unauthorized | //请求未经授权，这个状态代码必须和 WWW-Authenticate 报头域一起使用 |
| 403 Forbidden | //服务器收到请求，但是拒绝提供服务 |
| 404 Not Found | //请求资源不存在，eg: 输入了错误的 URL |
| 500 Internal Server Error | //服务器发生不可预期的错误 |
| 503 Server Unavailable | //服务器当前不能处理客户端的请求，一段时间后可能恢复正常 |

403Forbidden，有时候需要把django的csrf给注掉

20.接口开发

```
#view_if.py
from django.http import JsonResponse
from sign.models import Event, Guest
from django.core.exceptions import ValidationError, ObjectDoesNotExist #格式错误,和数据不存在
from django.db.utils import IntegrityError #手机号问题
import time
```

(1)添加发布会接口

```
def add_event(request):
    eid = request.POST.get('eid', '') # 发布会id
    name = request.POST.get('name', '') # 发布会标题
    limit = request.POST.get('limit', '') # 限制人数
    status = request.POST.get('status', '') # 状态
    address = request.POST.get('address', '') # 地址
    start_time = request.POST.get('start_time', '') # 发布会时间

    if eid == '' or name == '' or limit == '' or address == '' or start_time == '':
        return JsonResponse({'status': 10021, 'message': 'parameter error'})

    result = Event.objects.filter(id=eid)
    if result:
        return JsonResponse({'status': 10022, 'message': 'event id already exists'})
```

```

result = Event.objects.filter(name=name)
if result:
    return JsonResponse({'status':10023,'message':'event name already exists'})

if status == '':
    status = 1

try:
    Event.objects.create(id=eid,name=name,limit=limit,address=address,status=int(status),start_time=start_time)
except ValidationError:#格式错误
    error = 'start_time format error. It must be in YYYY-MM-DD HH:MM:SS format.'
    return JsonResponse({'status':10024,'message':error})

return JsonResponse({'status':200,'message':'add event success'})

```

#(2)发布会查询接口

```

def get_event_list(request):

    eid = request.GET.get('eid','') #发布会id
    name = request.GET.get('name','') #发布会名称

    if eid == '' and name == '':
        return JsonResponse({'status':10021,'message':'parameter error'})

    if eid != '':
        #字典里边嵌入字典
        event = {}
        try:
            result = Event.objects.get(id=eid)
        except ObjectDoesNotExist:
            return JsonResponse({'status':10022,'message':'query result is empty'})
        else:
            #这个是给后边的结果赋值
            event['name'] = result.name
            event['limit'] = result.limit
            event['status'] = result.status
            event['address'] = result.address
            event['start_time'] = result.start_time

            return JsonResponse({'status':200,'message':'success','data':event})

    if name != '':
        #python传递数组字典
        datas = []
        results = Event.objects.filter(name__contains=name)
        if results:

```

```

    for r in results:
        event = {}
        event['name'] = r.name
        event['limit'] = r.limit
        event['status'] = r.status
        event['address'] = r.address
        event['start_time'] = r.start_time
        datas.append(event)

    return JsonResponse({'status':200,'message':'success','data':datas})
else:
    return JsonResponse({'status':10022,'message':'query result is empty'})

```

#(3)添加嘉宾接口

```

def add_guest(request):
    eid = request.POST.get('eid','')#关联发布会id
    realname = request.POST.get('realname','') #姓名
    phone = request.POST.get('phone','')#手机号
    email = request.POST.get('email','')#邮箱
    #判断参数是否存在
    if eid == '' or realname == '' or phone == '':
        return JsonResponse({'status':10021,'message':'parameter error'})

    #判断是不是链接好了发布会的
    result = Event.objects.filter(id = eid)
    if not result:
        return JsonResponse({'status':10022,'message':'event id null'})

    #看下发布会的状态
    result = Event.objects.get(id = eid).status
    if not result:
        return JsonResponse({'status':10023,'message':'event status is not available'})

    #发布会限制人数
    event_limit = Event.objects.get(id = eid).limit #发布会限制人数
    #发布会已添加的嘉宾数
    guest_limit = len(Guest.objects.filter(event_id = eid))

    if guest_limit >= event_limit:
        return JsonResponse({'status':10024,'message':'event number is full'})

    #发布会时间

```

```

event_time = Event.objects.get(id = eid).start_time
#将发布会时间转为字符串，以点分开，取元组第一个,得到点以前的数据
etime = str(event_time).split('.')[0]

#将时间字符串转换成指定格式strptime()
timeArray = time.strptime(etime,'%Y-%m-%d %H:%M:%S')

#将开始时间转化为指定秒数mktime()
e_time = int(time.mktime(timeArray))

#当前时间，这个就是以秒数做单位的
now_time = str(time.time())

#去掉后边的小数点部分
ntime = now_time.split('.')[0]

#转化成整数
n_time = int(ntime)

if n_time >= e_time:
    return JsonResponse({'status':10025,'message':'event has
started','ntime':ntime,'e_time':e_time})

try:
    #将上边的数据都比对完了之后
    Guest.objects.create(realname = realname,phone = int(phone),email = email,sign =
0,event_id = int(eid))
except IntegrityError:
    return JsonResponse({'status':10026,'message':'the event guest phone number
repeat'})

return JsonResponse({'status':200,'message':'add guest success'})

```

#(4)嘉宾查询接口

```
def get_guest_list(request):
```

```

    eid = request.GET.get('eid','')#关联发布会id
    phone = request.GET.get('phone','')#嘉宾手机号
    #如果发布会id为空
    if eid == '':
        return JsonResponse({'status':10021,'message':'eid can not be empty'})
    #如果发布会id存在，手机为空,看能不能通过一个参数获取数据
    if eid != '' and phone == '':

```



```

datas = []
'''模糊查询'''
results = Guest.objects.filter(event_id = eid)
#如果列表存在
if results:
    for r in results:
        guest = {}

        guest['realname'] = r.realname
        guest['phone'] = r.phone
        guest['email'] = r.email
        guest['sign'] = r.sign

        datas.append(guest)
    return JsonResponse({'status':200,'message':'success','data':datas})
else:
    return JsonResponse({'status':10022,'message':'query result is empty'})
'''如果eid和phone都不为空'''
if eid != '' and phone != '':
    '''精确查询'''
    guest = {}

    try:
        result = Guest.objects.get(phone = phone,event_id = eid)
    except ObjectDoesNotExist:
        return JsonResponse({'status':10022,'message':'query result is empty'})
    else:
        guest['realname'] = result.realname
        guest['phone'] = result.phone
        guest['email'] = result.email
        guest['sign'] = result.sign
        return JsonResponse({'status':200, 'message':'success', 'data':guest})

```

#(5)嘉宾签到接口

```
def user_sign(request):
```

```

    eid = request.POST.get('eid','')#发布会id
    phone = request.POST.get('phone','')#嘉宾手机号
    if eid == '' or phone == '':
        return JsonResponse({'status':10021,'message':'parameter error'})
    #根据发布会id找发布会，找到的发布会时唯一的
    result = Event.objects.filter(id = eid)
    #如果对应的发布会不存在
    if not result:
        return JsonResponse({'status':10022,'message':'event id null'})
    #如果发布会存在再判断状态
    result = Event.objects.filter(id = eid).status

```

```

if not result:
    return JsonResponse({'status':10023,'message':'event status is not available'})

#开始时间转化为时间戳
event_time = Event.objects.get(id = eid).start_time
#将当前时间.前的部分转化成字符串
etime = str(event_time).split('.')[0]
#将字符串转化为标准时间
timeArray = time.strptime(etime,'%Y-%m-%d %H:%M:%S')
#再将标准时间转化为时间戳
e_time = int(time.mktime(timeArray))

#当前时间转化为时间戳
#获得当前时间,这个时间直接就是时间戳, 并转化为字符串
now_time = str(time.time())
ntime = now_time.split('.')[0]
n_time = int(ntime)

#如果当前时间大于等于发布会开始时间
if n_time >= e_time:
    return JsonResponse({'status':10024,'message':'event has started'})

#通过手机号去获取用户,手机号和用户是对应的, 如果用户不存在, 手机号就不存在
result = Guest.objects.filter(phone = phone)
if not result:
    return JsonResponse({'status':10025,'message':'user phone null'})

#这里可以得到用户
result = Guest.objects.filter(event_id = eid, phone = phone)
if not result:
    return JsonResponse({'status':10026,'message':'user did not participate in the conference'})

result = Guest.objects.filter(event_id = eid,phone = phone).sign
if result:
    return JsonResponse({'status':10027,'message':'user has sign in'})
else:
    Guest.objects.filter(event_id = eid,phone = phone).update(sign = '1')
    return JsonResponse({'status':200,'message':'sign success'})

```

21.配置接口路径

当所有接口都已经开发完成, 需要配置接口的访问路径。

打开.../guest/urls.py 文件，添加接口基本路径“/api/”：

```
from django.conf.urls import url, include
urlpatterns = [
    .....
    url(r'^api/', include('sign.urls', namespace="sign")),
]
```

创建.../sign/urls.py 文件，配置具体接口的二级路径。

urls.py

```
from django.conf.urls import url
from sign import views_if

urlpatterns = [
    # guest system interface:
    # ex : /api/add_event/
    url(r'^add_event/', views_if.add_event, name='add_event'),
    # ex : /api/add_guest/
    url(r'^add_guest/', views_if.add_guest, name='add_guest'),
    # ex : /api/get_event_list/
    url(r'^get_event_list/', views_if.get_event_list, name='get_event_list'),
    # ex : /api/get_guest_list/
    url(r'^get_guest_list/', views_if.get_guest_list, name='get_guest_list'),
    # ex : /api/user_sign/
    url(r'^user_sign/', views_if.user_sign, name='user_sign'),
]
```

22.接口的请求和测试

(1)请求

```
pre: pip3 install requests
```

```
test_get_eventlist.py ●
#test_get_eventlist.py
import requests
import json
# from importlib import reload
# import sys
# reload(sys)
# sys.setdefaultencoding( "utf-8" )

url = "http://127.0.0.1:8000/api/get_event_list/"
r = requests.get(url, params={'eid':'1'})
result = r.json()
# jsondata = json.dumps(result,ensure_ascii = False).encode('utf-8')
jsondata = json.dumps(result).encode('utf-8')
print(jsondata)

# .decode('unicode-escape')
# jsondata= json.dumps( dics, ensure_ascii = False, indent = 4 )
```

```
test_add_event.py ×
import requests

url = "http://127.0.0.1:8000/api/add_event/"
payload = {'eid':3,'name':'小米6','limit':2000,
'address':"北京水立方",'start_time':'2017-05-10 12:00:00'}

r = requests.post(url,data = payload)

result = r.json()

print(result)
```

(2)测试

如果是简单测试的话建议使用postman, poster等

如果压测使用jmeter

网上有教程, 上手很快

23.部署

这个后续添加