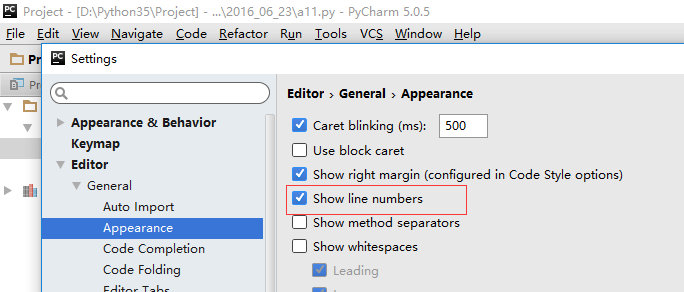
# pycharm

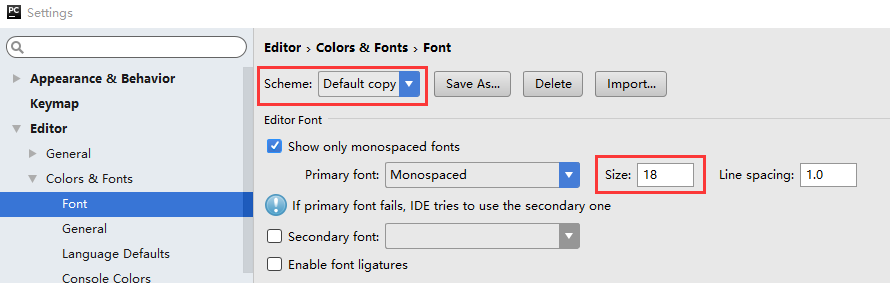
## 若干技巧

### 显示行数



### 字体大小

 Ide-setting  ——>Editor ——>Colors&Fonts ——> Font

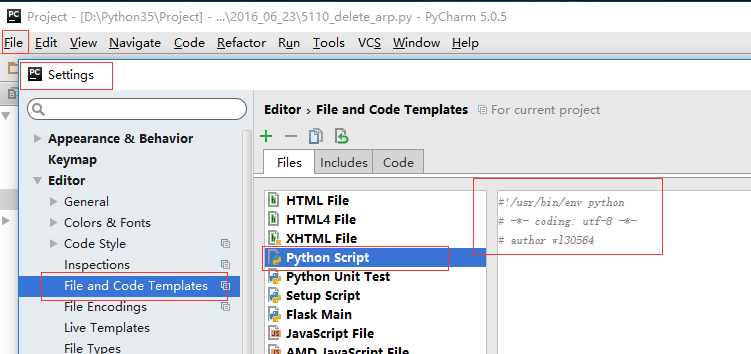


### Code Templates

#!/usr/bin/env python

# -\*- coding: utf-8 -\*-

# author wl30564

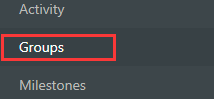


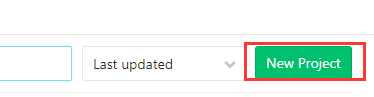
## 新建项目

1 先用pycharm 创建一个django项目

2 然后新建若干个APP

3 在公司的gitlab上面新建一个项目





4 然后把项目和gitlab关联起来

git init

git remote add origin [git@git.17usoft.com:it-ops/packetfence.git](mailto:git@git.17usoft.com:it-ops/packetfence.git)

# django

## virtualenv

### step1 install virtualenv

pip install virtualenv

# 创建一个虚拟环境

d:\>virtualenv testvir

Using base prefix 'd:\\program files\\python35'

New python executable in d:\testvir\Scripts\python.exe

Installing setuptools, pip, wheel...done.

进入虚拟环境

d:\testvir\Scripts>activate.bat

(testvir) d:\testvir\Scripts>pip list

离开虚拟环境

(testvir) d:\testvir\Scripts>deactivate.bat

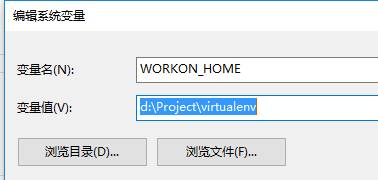
# 删除虚拟环境

直接删除虚拟环境所在的文件夹venv

### step install virtualenvwrapper-win

pip install virtualenvwrapper-win

# 设置安装路径



# 新建一个虚拟环境

mkvirtualenv env01

# 查看系统种虚拟环境

D:\testvir\Scripts>workon

# 进入虚拟环境

d:\testvir\Scripts>workon testvir2

## Getting Started With django

### step1: installing django

pip:

pip install Django

查看安装版本

import django

print(django.VERSION)

# (1, 11, 4, 'final', 0)

下载源码，进入根目录 执行python setup.py install

### step2: new django project

linux:

django-admin.py startproject ubuntu

windows:

pycharm

### step3: new app

python manage.py startapp blog

然后在setting INSTALLED\_APPS=[]中添加 app的名字

### step4: 创建数据库

djangp setting

DATABASES = {

'default': {

'ENGINE': 'django.db.backends.mysql',

'NAME': 'wl30564',

'USER': 'root',

'PASSWORD': 'Tc@891140',

'HOST': '10.101.26.64',

}

}

pip install mysqlclient-1.3.12-cp35-cp35m-win\_amd64.whl

然后在setting的同一层目录的\_\_init\_\_.py文件里面添加：

import pymysql

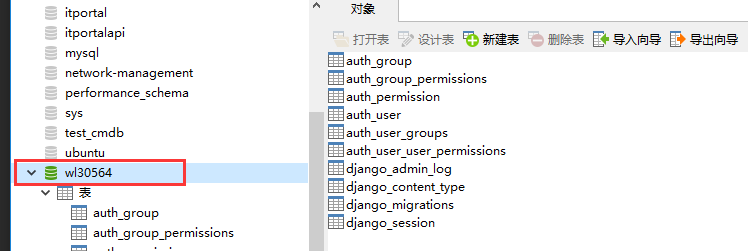
pymysql.install\_as\_MySQLdb()

在mysql中新建一个wl3056

最后同步一下：

python manage.py makemigrations

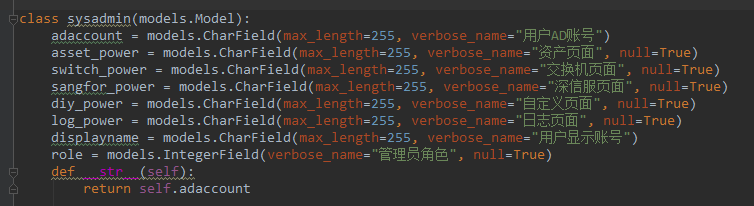
python manage.py migrate



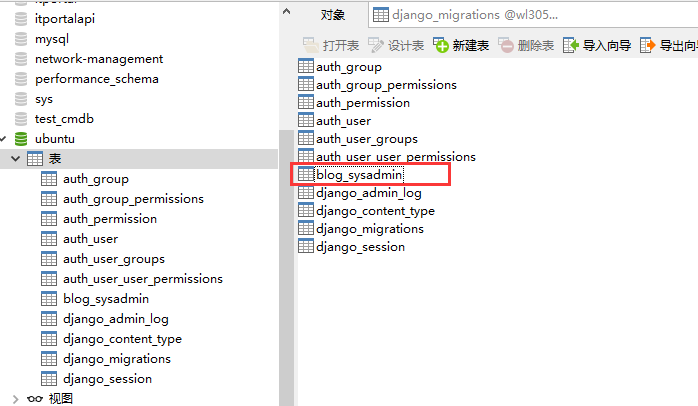
### 4.1 表操作

创建表：

在对应app的models中添加



同步一下

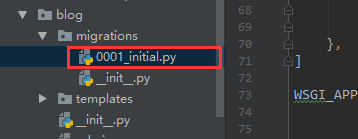


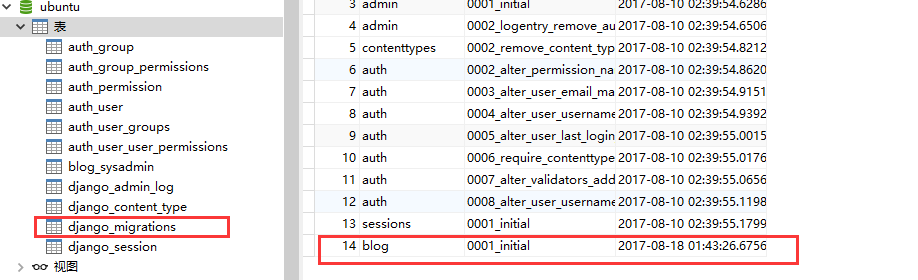
可以看到 表已经创建好了，

修改表：

先在models中修改，

删除





然后在同步一下 。

### step5: 使用开发服务器

python manage.py runserver 0.0.0.0:8000

### 6. 创建超级管理员

python manage.py createsuperuser

# 修改 用户密码可以用：

python manage.py changepassword username

#### 6.1 settings设置

1 把多个app 放到apps里面

右击 apps mark source root

sys.path.insert(0, os.path.join(BASE\_DIR, 'apps'))

2 设置中文和时区

LANGUAGE\_CODE = 'zh-Hans'

TIME\_ZONE = 'Asia/Shanghai'

USE\_TZ = False

### 7. 导出数据 导入数据

python manage.py dumpdata appname > appname.json

python manage.py loaddata appname.json

### 8. Django 项目环境终端

python manage.py shell

如果你安装了 bpython 或 ipython 会自动用它们的界面，推荐安装 bpython。

这个命令和 直接运行 python 或 bpython 进入 shell 的区别是：你可以在这个 shell 里面调用当前项目的 models.py 中的 API，对于操作数据，还有一些小测试非常方便。

### 9. 数据库命令行

python manage.py dbshell

Django 会自动进入在settings.py中设置的数据库，如果是 MySQL 或 postgreSQL,会要求输入数据库用户密码。

在这个终端可以执行数据库的SQL语句。如果您对SQL比较熟悉，可能喜欢这种方式。

### 10. 更多命令

终端上输入 python manage.py 可以看到详细的列表，在忘记子名称的时候特别有用。

## 视图与网址

### step1: 定义视图函数

在app目录中,把views.py打开

**from** django**.**http **import** HttpResponse

**def** index**(**request**):**

**return** HttpResponse**(**u'欢迎光临，同程大厦'**)**

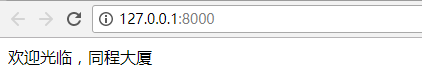
**from** blog **import** views **as** blog\_views

urlpatterns **=** **[**

url**(**r'^$'**,** blog\_views**.**index**),**

**]**

访问结果如下：



### step2: 视图与网址进阶

在网页上做加减法

1. 采用 /add/?a=4&b=5 这样GET方法进行

ubuntu.blog.views

**def** add**(**request**):**

a **=** request**.**GET**.**get**(**'a'**,** 0**)**

b **=** request**.**GET**.**get**(**'b'**,** 0**)**

c **=** int**(**a**)** **+** int**(**b**)**

**return** HttpResponse**(**c**)**

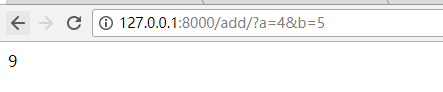
ubuntu.ubuntu.urls

urlpatterns **=** **[**

url**(**r'^add/$'**,** blog\_views**.**add**,** name**=**'add'**), # this line**

**]**

chrome



2. 采用 /add/3/4/ 这样的网址的方式

ubuntu.blog.views

**def** add2**(**request**,** a**,** b**):**

c **=** int**(**a**)** **+** int**(**b**)**

**return** HttpResponse**(**str**(**c**))**

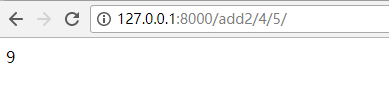
ubuntu.ubuntu.urls

urlpatterns **=** **[**

url**(**r'^add/(\d+)/(\d+)/$'**,** blog\_views**.**add2**,** name**=**'add2'**),**

**]**

chrome



### URL name详解

url**(**r'^add/$'**,** blog\_views**.**add**,** name**=**'add'**),**

**name =’add’ 是用来干嘛的呢？**

简单说，name 可以用于在 templates, models, views ……中得到对应的网址，相当于“给网址取了个名字”，只要这个名字不变，网址变了也能通过名字获取到。

## Django 模板(templates)

### 1 入门

前面我们都是用简单的 django.http.HttpResponse 来把内容显示到网页上，本节将讲解如何使用渲染模板的方法来显示内容。

1 写一个首页的视图

**from** django**.**shortcuts **import** render

**def** index**(**request**):**

**return** render**(**request**,** 'blog/home.html'**)**

2 在templates 新建一个blog文件夹写一个home.html

**<!**DOCTYPE html**>**

**<**html lang**=**"en"**>**

**<**head**>**

**<**meta charset**=**"UTF-8"**>**

**<**title**>**同程大厦**</**title**>**

**</**head**>**

**<**body**>**

欢迎光临自强学堂

**</**body**>**

**</**html**>**

3 写一个url

url**(**r'^$'**,** blog\_views**.**index**,** name**=**'index'**),**

4 显示



#### django中的超链接

href 后面是 目标地址

template中可以用 “{% url ‘app\_name:url\_name’ param %}”

ex:<a href="{% url 'blog:article\_page' article.id %}">{{ article.title }}</a>



### 2 扩展

网站模板的设计，一般的，我们做网站有一些通用的部分，比如 导航，底部，访问统计代码等等nav.html, bottom.html, tongji.html可以写一个 base.html 来包含这些通用文件（include)

### 3 晋级

Django模板中的循环，条件判断，常用的标签，过滤器的使用。

1. 列表，字典，类的实例的使用
2. 循环：迭代显示列表，字典等中的内容
3. 条件判断：判断是否显示该内容，比如判断是手机访问，还是电脑访问，给出不一样的代码。
4. 标签：for，if 这样的功能都是标签。
5. 过滤器：管道符号后面的功能，比如{{ var|length }}，求变量长度的 length 就是一个过滤器。

将一个或多个变量共享给多个网页或者所有网页使用，比如在网页上显示来访者的IP，这个可以使用 Django 上下文渲染器

#### step1 显示字符串

views.py

**def** home**(**request**):**

string **=** u'我在同程大厦'

**return** render**(**request**,** 'home.html'**,** **{**'string' **:** string**})**

视图中我们传递了一个字符串名称是 string 到模板 home.html

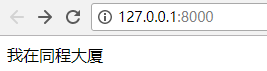
home.html

{{ string }}

url

url(r'^$', blog\_views.home, name='home'),

显示



#### step2 for循环 list显示

1 views:

**def** home**(**request**):**

list **=** **[**'html'**,** 'css'**,** 'jquery'**,** 'python'**,** 'django'**]**

**return** render**(**request**,** 'home.html'**,** **{**'list'**:** list**})**

2 home.html:

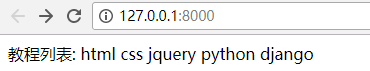
教程列表**:**

**{%** **for** i **in** list **%}**

**{{** i **}}**

**{%** endfor **%}**

3 显示



#### step3 显示字典

views:

**def** home**(**request**):**

info\_dict **=** **{**'site'**:** u'同程大厦'**,** 'content'**:** u'研发中心'**}**

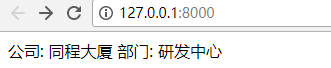
**return** render**(**request**,** 'home.html'**,** **{**'info\_dict'**:** info\_dict**})**

home.html

公司**:** **{{** info\_dict**.**site **}}**

部门**:** **{{** info\_dict**.**content **}}**

显示:



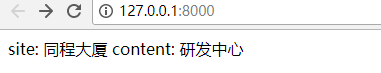
方法二：

home.html

**{%** **for** key**,** value **in** info\_dict**.**items **%}**

**{{** key **}}:** **{{** value **}}**

**{%** endfor **%}**



#### step4 条件判断 for循环

views:

**def** home**(**request**):**

list **=** map**(**str**,** range**(**100**))**

**return** render**(**request**,** 'home.html'**,** **{**'list'**:** list**})**

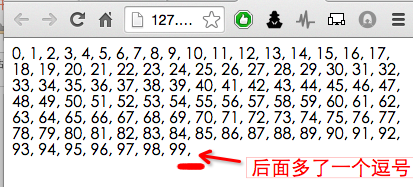
home.html

**{%** **for** item **in** list **%}**

**{{** item **}},**

**{%** endfor **%}**

显示：



改进：

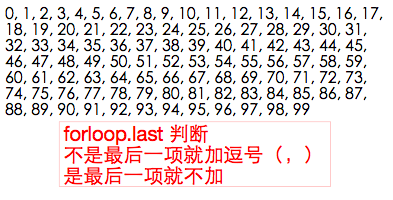
用变量 forloop.last 这个变量，如果是最后一项其为真，否则为假，更改如下

home.html

**{%** **for** item **in** list **%}**

**{{** item **}}{%** **if** **not** forloop**.**last **%},** **{%** endif **%}**

**{%** endfor **%}**



|  |  |
| --- | --- |
| **forloop.counter** | 索引从 1 开始算 |
| **forloop.counter0** | 索引从 0 开始算 |
| **forloop.revcounter** | 索引从最大长度到 1 |
| **forloop.revcounter0** | 索引从最大长度到 0 |
| **forloop.first** | 当遍历的元素为第一项时为真 |
| **forloop.last** | 当遍历的元素为最后一项时为真 |
| **forloop.parentloop** | 用在嵌套的 for 循环中，  获取上一层 for 循环的 forloop |

#### step5 模板上得到视图对应的网址

没看懂。

# views.py

def add(request, a, b):

    c = int(a) + int(b)

    return HttpResponse(str(c))

# urls.py

urlpatterns = patterns('',

    url(r'^add/(\d+)/(\d+)/$', 'app.views.add', name='add'),

)

# template html

{% url 'add' 4 5 %}

这样网址上就会显示出：/add/4/5/ 这个网址，假如我们以后修改 urls.py 中的

r'^jiafa/(\d+)/(\d+)/$'

这样，我们不需要再次修改模板，当再次访问的时候，网址会自动变成 /jiafa/4/5/

#### step6 模板中的逻辑操作

==, !=, >=, <=, >, < 这些比较都可以在模板中使用

{% if var >= 90 %}

成绩优秀，自强学堂你没少去吧！学得不错

{% elif var >= 80 %}

成绩良好

{% elif var >= 70 %}

成绩一般

{% elif var >= 60 %}

需要努力

{% else %}

不及格啊，大哥！多去自强学堂学习啊！

{% endif %}

**注意：比较符号前后必须有至少一个空格！**

and, or, not, in, not in 也可以在模板中使用

假如我们判断 num 是不是在 0 到 100 之间：

{% if num <= 100 and num >= 0 %}

num在0到100之间

{% else %}

数值不在范围之内！

{% endif %}

#### step7 模板中 获取当前网址 用户

推荐用 [**render**](https://docs.djangoproject.com/en/dev/topics/http/shortcuts/#render) 而不是用 [render\_to\_response](https://docs.djangoproject.com/en/dev/topics/http/shortcuts/#django.shortcuts.render_to_response)

render()方法是render\_to\_response的一个崭新的快捷方式

return render(request, 'blog\_add.html',locals())

return render\_to\_response('blog\_add.html',locals())

HttpReponse返回的直接是内容

return HttpResponse(u'欢迎光临，同程大厦')

**获取当前用户：**

{{ request.user }}

如果登陆就显示内容，不登陆就不显示内容：

{% if request.user.is\_authenticated %}

    {{ request.user.username }}，您好！

{% else %}

    请登陆，这里放登陆链接

{% endif %}

**获取当前网址：**

{{ request.path }}

**获取当前 GET 参数：**

{{ request.GET.urlencode }}

**合并到一起用的一个例子：**

<a href="{{ request.path }}?{{ request.GET.urlencode }}&delete=1">当前网址加参数 delete</a>

比如我们可以判断 delete 参数是不是 1 来删除当前的页面内容。

## 模型(数据库)

### step1 修改models

**class** **Person(**models**.**Model**):**

name **=** models**.**CharField**(**max\_length**=**30**)**

age **=** models**.**IntegerField**()**

**def** \_\_str\_\_**(**self**):**

**return** self**.**name

### step2 创建数据表

python manage.py makemigrations

python manage.py migrate

### step3 数据表更改

我们设计数据库的时候，早期设计完后，后期会发现不完善，要对数据表进行更改

### step4 使用QuerySet API

**from** people**.**models **import** Person

# 新建一条数据

Person**.**objects**.**create**(**name**=**"WeizhongTu"**,** age**=**24**)**

# 查询一条数据

Person**.**objects**.**get**(**name**=**"WeizhongTu"**)**

note:

在写models的时候 如果没有

def \_\_str\_\_(self):

return self.name

Person.objects.get 查询的结果是: <QuerySet [<Person: Person object>]>

启用了

def \_\_str\_\_(self):

return self.name

Person.objects.get 查询的结果是: <QuerySet [<Person: WZT>]>

#### 新建

1 Person.objects.create(name=name,age=age)

2 Person.objects.get\_or\_create(name="WZT", age=23)

(<Person: Person object>, False)

防止重复很好的方法，但是速度要相对慢些，返回的是一个元组，(object, True/False)，创建时返回 True, 已经存在时返回 False

#### 查询

**GET**

Person.objects.all() # 查询所有

Person.objects.all()[:10] 切片操作，获取10个人，不支持负索引，切片可以节约内存，不支持负索引，后面有相应解决办法，第7条

Person.objects.get(name="WeizhongTu") # 名称为 WeizhongTu 的一条，多条会报错

**filter**

**查询数量：**

directorcounts = flow.objects.using('itportal').filter(director=adaccount).filter(flowstatus=0).count()

1. Person.objects.filter(name="abc")  # 等于Person.objects.filter(name\_\_exact="abc") 名称严格等于 "abc" 的人
2. Person.objects.filter(name\_\_iexact="abc")  # 名称为 abc 但是不区分大小写，可以找到 ABC, Abc, aBC，这些都符合条件
3. Person.objects.filter(name\_\_contains="abc")  # 名称中包含 "abc"的人
4. Person.objects.filter(name\_\_icontains="abc")  #名称中包含 "abc"，且abc不区分大小写
5. Person.objects.filter(name\_\_regex="^abc")  # 正则表达式查询
6. Person.objects.filter(name\_\_iregex="^abc")  # 正则表达式不区分大小写

filter是找出满足条件的，当然也有排除符合某条件的

1. Person.objects.exclude(name\_\_contains="WZ")  # 排除包含 WZ 的Person对象
2. Person.objects.filter(name\_\_contains="abc").exclude(age=23)  # 找出名称含有abc, 但是排除年龄是23岁的

查询结果排序

Author.objects.all().order\_by('name')

Author.objects.all().order\_by('-name') # 在 column name 前加一个负号，可以实现倒序

#### 支持链式查询

Author.objects.filter(name\_\_contains="WeizhongTu").filter(email="tuweizhong@163.com")

Author.objects.filter(name\_\_contains="Wei").exclude(email="tuweizhong@163.com")

# 找出名称含有abc, 但是排除年龄是23岁的

Person.objects.filter(name\_\_contains="abc").exclude(age=23)

不支持负索引

Person.objects.all()[:10] 切片操作，前10条

Person.objects.all()[-10:] 会报错！！！

# 1. 使用 reverse() 解决

Person.objects.all().reverse()[:2] # 最后两条

Person.objects.all().reverse()[0] # 最后一条

# 2. 使用 order\_by，在栏目名（column name）前加一个负号

Author.objects.order\_by('-id')[:20] # id最大的20条

#### 去重

QuerySet 重复的问题，使用 .distinct() 去重

qs1 = Pathway.objects.filter(label\_\_name='x')

qs2 = Pathway.objects.filter(reaction\_\_name='A + B >> C')

qs3 = Pathway.objects.filter(inputer\_\_name='WeizhongTu')

# 合并到一起

qs = qs1 | qs2 | qs3

这个时候就有可能出现重复的

# 去重方法

qs = qs.distinct()

#### 删除

Person.objects.filter(name\_\_contains="abc").delete() # 删除名称中包含 "abc"的人

#### 更新

**批量更新 适用于 .all() .filter() .exclude() 等后面**

Person.objects.filter(name\_\_contains="abc").update(name='xxx') # 名称中包含 "abc"的人 都改成 xxx

Person.objects.all().delete() # 删除所有 Person 记录

### step5 QuerySet 进阶

#### djano使用2个数据库

1 在setting中添加第二个数据库

DATABASES = {  
 'default': {  
 'ENGINE': 'django.db.backends.sqlite3',  
 'NAME': os.path.join(BASE\_DIR, 'db.sqlite3'),  
 },  
  
 # 使用第二个数据库, step1 添加databases  
 'pf': {  
 'ENGINE': 'django.db.backends.mysql',  
 'NAME': 'pf',  
 'USER': 'root',  
 'PASSWORD': 'Wbb601378656',  
 'HOST': '10.101.26.207',  
 'PORT': '3306'  
 }  
}

2 添加mapping

# 'app\_name':'database\_name',  
DATABASE\_APPS\_MAPPING = {'apps.net': 'pf'}

3 在models中创建表项

db\_table = "iplog"

app\_label = "apps.net"

这2个记得写上 不然查询有问题。

class Iplog(models.Model):  
 mac = models.CharField(max\_length=17, verbose\_name=u'用户mac地址')  
 ip = models.CharField(max\_length=45, verbose\_name=u'IP地址', primary\_key=True)  
 start\_time = models.DateTimeField(verbose\_name=u'开始时间')  
 end\_time = models.DateTimeField(default="0000-00-00 00:00:00", verbose\_name=u'结束时间')  
  
 class Meta:  
 verbose\_name = u'IP'  
 verbose\_name\_plural = verbose\_name  
 db\_table = "iplog"  
 app\_label = "apps.net"  
  
 def \_\_str\_\_(self):  
 return self.ip

## xadmin

### step1 install

for linux :

pip install xadmin

for windows :

pip install git+git://github.com/sshwsfc/xadmin.git

pip install django-import-export

### step2配置xadmin

settings.py

INSTALLED\_APPS = [

'xadmin',

'crispy\_forms',

]

urls.py

import xadmin

urlpatterns = [

url(r'^xadmin/', xadmin.site.urls),

### step3 同步数据库

makemigration

migrate

### step4 访问

用户名 在django项目中 createsuperuser

http://127.0.0.1:8000/xadmin