This post was republished to 汪汪家园 at 0:14:21 2017/2/23

Zakkabag

# 微信公众号及微网站

## 注册登录

### 普通登录

用户通过手机中微信登录网站

### 第三方登录

可以通过微信直接登录微网站，减少登录流程



**配置**

首先登录微信测试号，修改授权

|  |  |
| --- | --- |
| 网页帐号 | [网页授权获取用户基本信息](http://mp.weixin.qq.com/wiki/17/c0f37d5704f0b64713d5d2c37b468d75.html) |

127.0.0.1:8000

确保” 微信web开发者工具”已安装，这个对微信调试非常有帮助

#### 登录入口

首先提供微信登录入口，需要

1. 在导航条上添加微信登录图标，具体图标可以从bootstrap上得到
2. 在URL添加入口

navbar.html

<li><a href=**'{% url "wechatlogin" %}'**><i class=**"fa fa-weixin"** aria-hidden=**"true"**></i></a></li>

zakkabag.urls.py

urlpatterns **=** patterns**(**''**,**

url**(**r'^wechatlogin/$'**,** 'wechat.views.wechatlogin'**,** name**=**'wechatlogin'**),**

url**(**r'^logout/$'**,** 'newsletter.views.logout'**,** name**=**'logout'**),**

url**(**r'^login/$'**,** 'newsletter.views.login'**,** name**=**'login'**),**

#### 微信授权

接下来将在wechatlogin里面实现授权功能

关于微信登录的信息可参考：[**微信公众平台开发者文档**](https://mp.weixin.qq.com/wiki)

具体的实现我们可参考一个python library : [python-weixin](https://pypi.python.org/pypi/python-weixin)

登录成功后，它会重定向到login URL

**from** weixin**.**client **import** WeixinMpAPI

**def** wechatlogin**(**request**):**

REDIRECT\_URI **=** request**.**build\_absolute\_uri**(**'/'**).**strip**(**"/"**)** **+** reverse**(**"login"**,** kwargs**={})**

api **=** WeixinMpAPI**(**appid**=**APP\_ID**,** app\_secret**=**APP\_SECRET**,**redirect\_uri**=**REDIRECT\_URI**)**

redirect\_uri **=** api**.**get\_authorize\_login\_url**(**scope**=(**"snsapi\_userinfo"**,))**

**return** redirect**(**redirect\_uri**)**

#### 继续登录

获得weixin授权之后，接下来再里newsletter.views.login实现登录，中间会用到两个django函数authenticate和auth.login。

如果用户授权失败，则重定向到登录界面

**def** login**(**request**):**

REDIRECT\_URI **=** request**.**POST**.**get**(**'next'**,** request**.**GET**.**get**(**'next'**,** reverse**(**"home"**,** kwargs**={})))** #template指定的重定向地址

**if** request**.**method **==** 'GET'**:**

code **=** request**.**GET**.**get**(**'code'**)**

**if** code**:**

redirect\_to **=** "http://%s%s" **%** **(**request**.**META**[**'HTTP\_HOST'**],** reverse**(**"home"**,** kwargs**={}))**#登录成功后的登录地址

api **=** WeixinMpAPI**(**appid**=**APP\_ID**,**

app\_secret**=**APP\_SECRET**,**

redirect\_uri**=**redirect\_to**)**

auth\_info **=** api**.**exchange\_code\_for\_access\_token**(**code**=**code**)**

api **=** WeixinMpAPI**(**access\_token**=**auth\_info**[**'access\_token'**])**

api\_user **=** api**.**user**(**openid**=**auth\_info**[**'openid'**])**

user **=** authenticate**(**request**=**request**,** user**=**api\_user**)**

**if** user**:** #如果用户授权失败，则重定向到登录界面

auth**.**login**(**request**,** user**)**

**return** redirect**(**redirect\_to**)**

**return** redirect**(**reverse**(**"auth\_login"**,** kwargs**={}))**

**return** auth\_views**.**login**(**request**,** redirect\_field\_name**=**REDIRECT\_URI**,** extra\_context**=None)**

特别注意的是，这边登录用了authenticate函数，参数第二个变成了user，这个其实已经对登录的后台进行了扩展，具体可参考章节“用户扩展”

登录成功后需要在templates.navbar.html修改图标为已登录状态

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

{% if request.user.account\_type == 'wechat' %}

<li><a href='{{request.user.get\_absolute\_url}}'}>{{request.user.get\_wechatprofile}}</a></li>

<li><a href="{% url 'logout' %}">Logout</a></li>

{% else %}

<li><a href="{{request.user.get\_absolute\_url}}">{{request.user}}</a></li>

<li><a href="{% url 'logout' %}">Logout</a></li>

{% endif %}

{% elif request.wechat %}

<li><a href="{{request.wechat.get\_absolute\_url}}">{{request.wechat}}</a></li>

<li><a href="{% url 'logout' %}">Logout</a></li>

{% else %}

<li><a href="{% url 'registration\_register' %}">Register</a></li>

<li><a href='{% url "wechatlogin" %}'><i class="fa fa-weixin" aria-hidden="true"></i></a></li>

{% endif %}

#### 退出登录

在newsletter.views.logout实现注销，会调用系统函数auth.logout。

**def** logout**(**request**):**

**try:**

**del** request**.**session**[**'wechat\_id'**]**

**except:**

**pass**

auth**.**logout**(**request**)**

**return** redirect**(**reverse**(**"home"**,** kwargs**={}))**

### 用户注册

用户通过手机注册成为会员，推荐使用手机号注册保证用户的真实性和唯一性，收集准确的会员资料，为后期营销增加有效的用户量

### 忘记密码

可通过忘记密码模块可找回帐号，保证用户信息和积分不会丢失，增加用户体验

## 个人中心

### 个人信息展示

展示头像用户名等级及其它用户信息

创建APP，personalcenter

### 上传头像

通过网站可以修改头像

### 我的积分

可以显示用户当前积分，如果需要冻结积分，也可显示冻结积分和可使用积分。还可以显示积分明细，纪录积分赚取花费情况

### 会员等级

可以根据捐衣次数或者兑换次数，或者消费积分的累积，享受不同的折扣兑换，增加用户对网站的粘性

### 我的捐赠

可以展示我捐赠衣服的次数，时间，上传的照片等其它信息

### 我的订单

显示我使用积分商城的兑换情况

## 设置

### 个人资料修改

修改个人信息资料，包含会员性别，生日，昵称等

### 密码修改

修改登录密码

### 意见反馈

给网站提出意见

### 注销登录

注销当前用户登录信息

## 活动消息

### 动态消息展示

可以显示后台发送的动态消息等

### 轮播图

轮播切换广告图片，可以放自己活动也可以作为广告位营销

### 活动专区

展示活动内容

## 捐衣模块

### 拍照上传

捐赠衣服前可以调用手机摄像头功能拍照上传

### 捐衣信息填报

填写捐赠件数等其它信息，还可以选择具体哪个捐赠衣服箱，方便工作人员审核和统计

## 众筹模块

### 众筹列表

可以列表展示众筹信息

### 众筹分类

可以根据大病救助，助学，创业分类展示众筹信息

### 众筹详情

可以展示众筹内容的展示，包含众筹的标题，内容以及图片和详情等具体信息

### 筹款动态

可以显示筹款的动态信息

### 众筹评论

对本条众筹可以添加评论信息

### 众筹收藏

可以收藏众筹信息，方便之后可以快速找到

### 众筹分享

可以把众筹信息，分享到主流的第三方平台上比如微信，朋友圈，微博，qq等

### 在线支付

可以通过在线支付支持爱心众筹，微网站支持微信一种支付方式

### 举报

如果发现不是真实的众筹信息，可以通过举报按钮提交举报，工作人员可以后台审核举报信息

### 发布众筹

可以通过平台发起大病救助，扶贫助学，支持创业等众筹项目，包含金额，用途，截止日期，标题，详情，照片，身份证信息等资料

## 积分商城

### 商品列表

展示积分商城的商品，包含图片，标题，所需兑换积分等信息

### 商品分类

可以根据分类展示不同的兑换商品

### 商品详情

展示单个积分商城商品的详细信息

### 积分兑换

通过积分兑换模块兑换所选择的商品，自动扣取积分

### 积分兑换规则

可以通过后台配置积分规则，前台此模块显示兑换的规则

# 公益平台管理系统（PC Web）

## 会员管理

### 会员认证管理

会员认证管理是对于会员的认证信息进行管理，可调整会员的认证方式包括手机认证、邮箱认证等其他第三方认证，关联微信openid,同时对会员的认证相关信息进行维护。

### 会员资料管理

会员资料管理是对会员的个人资料进行管理，可查看会员相关资料信息包括姓名、性别、手机、注册时间、邮箱、生日、剩余积分、等信息，可通过姓名、性别、手机号、等信息进行查询。提供新增会员，并根据需要进行查看详情、修改、删除功能。

### 会员积分管理

会员积分管理是对会员积分，兑换相关业务进行管理，可记录回溯兑换业务信息，包括手机号码、会员名称、兑换时间、兑换金额、本次积分、兑换商品、办理渠道、结算方式等，可通过电话以及选择相应开始时间到结束时间进行查询；可支撑会员的退积分业务处理。如果客户有ERP系统，各笔消费相关业务操作记录券可通过接口同步至ERP系统。

### 会员捐衣管理

对于会员捐衣纪录进行管理

### 会员等级管理

会员等级是可以新增会员等级，设置会员等级名称，消费金额，演出折扣，配置会员权益说明的功能。

### 会员协议管理

对线上平台出现的会员协议内容进行管理，可编辑修改会员协议内容。

## 微网站内容管理

### 微网站轮播图管理

对微网站轮播图进行新增，修改，删除，更换位置等管理

### 微网站活动管理

对微网站活动列表以及内容进行新增，修改，删除等管理

## app内容管理

### app轮播图管理

对app轮播图进行新增，修改，删除，更换位置等管理

### app活动管理

对app活动列表以及内容进行新增，修改，删除等管理

## 消息推送

### 全局消息推送

可以通过后台给全部会员发布消息

### 分组会员消息推送

可以根据对组员分类推送不同消息，比如对参加同个众筹的会员推送众筹的进度情况

## 众筹管理

### 官方发起众筹

官方发起的众筹的可以通过管理后台发布

### 众筹管理

对于平台上用户发起的众筹进度进行监控管理

### 审核管理

对用户提交的的众筹信息进行审核和对外发布，可以进行删除和管理

### 评论管理

管理对众筹发布的评论信息

### 举报管理

对于用户举报的众筹信息进行评估监控可以对其核实，删除等管理

### 支付管理

管理用户对众筹投入的资金，管控到期众筹的资金发放

## 商城管理

### 商品推荐管理

对积分商城推荐的商品进行推荐管理，可以更换推荐的位置，删除推荐等

### 商品上下架管理

对积分商城的商品进行上架和下架的维护和库存管理

### 商品详情管理

对金粉商城的商品进行商品内容，图片，详细信息的内容管理

### 订单管理

管理积分兑换的订单进行管理

## CBV

<https://docs.djangoproject.com/en/1.8/ref/class-based-views/generic-display/#detailview>

<https://docs.djangoproject.com/en/1.8/ref/class-based-views/generic-display/#listview>

[https://docs.djangoproject.com/en/1.8/ref/class-based-views/base/#view](https://docs.djangoproject.com/en/1.8/ref/class-based-views/base/" \l "view)

[https://docs.djangoproject.com/en/1.8/ref/class-based-views/generic-editing/#createview](https://docs.djangoproject.com/en/1.8/ref/class-based-views/generic-editing/" \l "createview)

<https://docs.djangoproject.com/en/1.8/ref/class-based-views/generic-editing/#formview>

<https://docs.djangoproject.com/en/1.8/ref/class-based-views/mixins-single-object/#django.views.generic.detail.SingleObjectMixin>

<https://docs.djangoproject.com/en/1.8/ref/class-based-views/mixins-editing/#formmixin>

DetailView

---- SingleObjectTemplateResponseMixin

-------- TemplateResponseMixin

---- BaseDetailView

-------- SingleObjectMixin

------------ ContextMixin

-------- View

ListView

---- SingleObjectTemplateResponseMixin

-------- TemplateResponseMixin

---- BaseListView

-------- MultipleObjectMixin

------------ ContextMixin

-------- View

CreateView

---- SingleObjectTemplateResponseMixin

-------- TemplateResponseMixin

---- BaseCreateView

-------- ModelFormMixin

------------ FormMixin

------------ SingleObjectMixin

----------------ContextMixin

-------- ProcessFormView

------------View

FormView

---- SingleObjectTemplateResponseMixin

-------- TemplateResponseMixin

---- BaseFormView

-------- FormMixin

-------- ProcessFormView

------------ View

FormMixin

---- FormMixinBase

---- ContextMixin

008 Product Detail View

011 ListView

039 Add, Update, Remove Carts in CBV

041 Render Cart View

092 Checkout from FromMixin

095 User Address View

100 Crafting the Order Part 3

105 Order List

**from** django**.**views**.**generic**.**edit **import** FormMixin

### URL

View.as\_view()会调用View.dispatch函数。

**from** **.**views **import** ProductDetailView**,** ProductListView

urlpatterns **=** **[**

url**(**r'^(?P<pk>\d+)/$'**,** ProductDetailView**.**as\_view**(),** name**=**'product\_detail'**),**

url**(**r'^$'**,** ProductListView**.**as\_view**(),** name**=**'products'**),**

**]**

**from** django**.**core**.**urlresolvers **import** reverse

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

**def** get\_absolute\_url**(**self**):**

**return** reverse**(**"product\_detail"**,** kwargs**={**"pk"**:** self**.**pk**})**

Products/templates/products/product\_list.html

<td><a href="/products/{{object.pk}}/">{{object.title}}</a></td>

<td><a href="{% url 'product\_detail' pk=object.pk %}">{{object.title}}</a></td>

<td><a href="{{ object.get\_absolute\_url }}">{{object.title}}</a></td>

### context

**from** django**.**views**.**generic**.**list **import** ListView

**from** django**.**utils **import** timezone

**class** **ProductListView(**ListView**):**

model **=** Product

**def** get\_context\_data**(**self**,** **\***args**,** **\*\***kwargs**):**

context **=** super**(**ProductListView**,** self**).**get\_context\_data**(\***args**,** **\*\***kwargs**)**

context**[**"now"**]** **=** timezone**.**now**()**

**return** context

### ModelManager

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

objects **=** ProductManager**()**

## 图片上传

在表单中加入[enctype](http://www.w3school.com.cn/tags/att_form_enctype.asp)="multipart/form-data；如果不加enctype="multipart/form-data：得不到图片信息，只有路径

FILES是个字典，它包含每个FileField的键 （或者ImageField，FileField的子类）。这样的话就可以用request.FILES['File']来存放表单中的这些数据了。

注意request.FILES只有在请求方法为POST，并且发送请求的<form>拥有enctype="multipart/form-data属性时，才会包含数据。否则request.FILES为空

products/forms.py

class ProductImageForm(forms.ModelForm):

class Meta:

model = ProductImage

fields = [

'image',

]

products/templates/products/product\_create.html

-<form method='POST' action=''>{% csrf\_token %}

+<form method='POST' enctype="multipart/form-data" action=''>{% csrf\_token %}

{{ form|crispy }}

+{{image\_form.as\_p}}

<input class = 'btn btn-primary' type='submit' value='Submit' />

</form>

products/views.py

class ProductCreateView(CreateView):

template\_name = 'products/product\_create.html'

form\_class = ProductCreateForm

success\_url = "/products"

def get\_context\_data(self, \*args, \*\*kwargs):

context = super(ProductCreateView, self).get\_context\_data(\*args, \*\*kwargs)

context["image\_form"] = ProductImageForm()

return context

def post(self, request, \*args, \*\*kwargs):

postresult = super(ProductCreateView, self).post(request, \*args, \*\*kwargs)

if 0: #方法1

filename=request.FILES['image']

from PIL import Image

if filename:

img=Image.open(filename)

title = self.object.title

slug = slugify(title)

basename, file\_extension = filename.name.split(".")

new\_filename = "%s-%s.%s" %(slug, self.object.id, file\_extension)

from django.conf import settings

import os

photoname = os.path.join("products", slug, new\_filename)

photopath = os.path.join(settings.MEDIA\_ROOT, "products", slug)

if not os.path.exists(photopath):

os.makedirs(photopath)

img.save(os.path.join(settings.MEDIA\_ROOT, photoname))

ProductImage.objects.create(product = self.object,

image = photoname)

# BELOW ALSO WORKS 方法2

else:

imageForm = ProductImageForm(request.POST, request.FILES)

if imageForm.is\_valid():

productImage = imageForm.save(commit=False)

productImage**.**product **=** self**.**object

productImage**.**save**()**

**return** postresult

**return** postresult

<https://docs.djangoproject.com/en/1.2/topics/http/file-uploads/>

[django上传图片](http://www.cnblogs.com/RoundGirl/p/5395658.html)

[python image模块](http://www.cnblogs.com/DjangoBlog/p/3557744.html)

[Python创建目录文件夹](http://www.qttc.net/201209207.html)

[[豆瓣]浅论Django文件上传](https://www.douban.com/group/topic/21131470/)

[Django Ajax 文件上传及进度显示](http://onlypython.group.iteye.com/group/wiki/1517-django-ajax-file-uploads-and-progress-display)

[Ajax 和 jQuery 实现进度条＋上传文件到Django](http://blog.csdn.net/dimply/article/details/42425373)

[ajax异步上传图片 ajax 异步上传带进度条视频并提取缩略图](http://www.alonely.com.cn/Ajax/20161025/58632.html)

[Django and AJAX image uploads](https://blog.mozilla.org/webdev/2010/09/17/django-and-ajax-image-uploads/) https://github.com/pcraciunoiu/kitsune/

[Django image upload](http://cloudinary.com/documentation/django_image_upload)

[jQuery File Upload Demo](https://blueimp.github.io/jQuery-File-Upload/)

## 用户扩展

用户扩展包括以下几种方法

* 直接修改django 源码
* 把django 的user以及认证部分的源代码拷贝到自己的app下面
* OneToOneField扩展
* Profile Model (1.5之后不再支持)
* AbstractBaseUser, PermissionsMixin开始派生出一个自定用户Model，并且实现自定义的BaseUserManager (1.5之后)
* ProfileBase(type)
* 继承User

### 指定authentication backend

Django维护了一个“authentication backends”列表，通过它在授权。Django会遍历所有的backends直到授权接收。

authentication backends在settings里的[AUTHENTICATION\_BACKENDS](https://docs.djangoproject.com/en/dev/ref/settings/#std:setting-AUTHENTICATION_BACKENDS)指定，可以是任意路径，默认的backends是

['django.contrib.auth.backends.ModelBackend']

[AUTHENTICATION\_BACKENDS](https://docs.djangoproject.com/en/dev/ref/settings/#std:setting-AUTHENTICATION_BACKENDS)的顺序是有影响的，如果多个backends的用户名和密码都能验证通过，django在第一次匹配后就会停止。

如果backend抛出[PermissionDenied](https://docs.djangoproject.com/en/dev/ref/exceptions/#django.core.exceptions.PermissionDenied)异常，授权检查会立即停止，不会再检查后面的backend。

注意：如果授权成功，django会把这个授权方式保存到session里，session周期里的下一次接入还是用这种方式。如果要强迫用不同方法授权，一个简单的方法是调用Session.objects.all().delete().

*Settings.py*

AUTHENTICATION\_BACKENDS **=** **(**

'personalcenter.backends.auth.WechatBackend'**,**

'django.contrib.auth.backends.ModelBackend'**,**

**)**

AUTH\_USER\_MODEL **=** 'personalcenter.MyUser'

### 实现authentication backend

必须实现的方法

get\_user(user\_id)

authenticate(request, \*\*credentials)

不知道为什么？username登录时的request.user是有值的，但是wechat登录永远是anonymous，这个从一开始的render就开始了

\_cached\_user为AnonymousUser: AnonymousUser

问题查清楚了，下面这个wechat auth backend函数没写好，之前返回None，这个函数在eclipse上打断点也进不去不知道为什么

好像也不是这个问题，突然就好了

personalcenter.auth.py

class WechatBackend(object):

def get\_user(self, user\_id):

try:

return UserModel.objects.get(pk=user\_id)

except UserModel.DoesNotExist:

return None

**def** authenticate**(**self**,** request**,** user**):**

obj **=** **None**

profile **=** **None**

**try:**

profile **=** WechatUserProfile**.**objects**.**get**(**openid **=** user**[**'openid'**])**

**if** profile**:**

obj **=** profile**.**user

**except** WechatUserProfile**.**DoesNotExist**:**

profile **=** WechatUserProfile**.**objects**.**create**(**

user **=** **None,** openid **=** user**[**'openid'**])**

profile**.**unionid **=** user**[**'unionid'**]**

profile**.**city **=** user**[**'city'**]**

profile**.**country **=** user**[**'country'**]**

profile**.**language **=** user**[**'language'**]**

**if** 1 **==** user**[**'sex'**]:**

profile**.**sex **=** 'male'

**else:**

profile**.**sex **=** 'female'

profile**.**nickname **=** user**[**'nickname'**]**

profile**.**headimgurl **=** user**[**'headimgurl'**]**

profile**.**save**()**

request**.**session**[**'wechat\_id'**]** **=** profile**.**id

request**.**user **=** obj

**return** obj

### wechat\_id的保存方法

Workaround方法 – 可能已解决user为空的情况

class openidmiddleware():

def process\_request(self, request):

wechat\_id = request.session.get("wechat\_id")

if wechat\_id:

UserModel = get\_user\_model()

try:

request.user = UserModel.objects.get(pk=wechat\_id)

except UserModel.DoesNotExist:

pass

class WechatBackend(object):

def authenticate(self, request, code, app\_id=None, app\_secret=None, redirect\_uri=None):

request.session['wechat\_id'] = obj.id

return None

### 绑定WechatUser和User

personalcenter.urls

urlpatterns **=** **[**

url**(**r'^linktowechat$'**,** accountlinktowechat**,** name**=**'link\_to\_wechat'**),**

**def** accountlinktowechat**(**request**):**

user **=** auth**.**get\_user**(**request**)**

wechat\_id **=** request**.**session**.**get**(**"wechat\_id"**)**

wechat **=** **None**

**if** wechat\_id**:**

**try:**

wechat **=** WechatUserProfile**.**objects**.**get**(**pk**=**wechat\_id**)**

wechat**.**user **=** user #这儿将WechatUser和User绑定起来

wechat**.**save**()**

**return** redirect**(**reverse**(**"home"**,** kwargs**={}))**

**except:**

**pass**

**return** redirect**(**reverse**(**"home"**,** kwargs**={}))**

#### 激活成功后绑定当前微信用户

templates.registration.activation\_complete.html

{% block content %}

<p class = "lead">

{% trans "Your account is now activated." %}

{% if not user.is\_authenticated %}

{% trans "You can log in." %}

{% else %}

<a href='{% url "link\_to\_wechat" %}'>Link to wechat</a>

{% endif %}

</p>

{% endblock %}

#### 微信登录后绑定当前登录用户

request middleware里面把wechat信息添加进去，这样在template里面能判断微信用户和普通用户是否已绑定

这儿额外在request里面加了一个变量wechat，否则就要给registration添加额外的context (第三方库)

**from** django**.**contrib**.**auth **import** get\_user\_model

**from** personalcenter**.**models **import** WechatUserProfile

UserModel **=** get\_user\_model**()**

**class** **openidmiddleware():**

**def** process\_request**(**self**,** request**):**

**if** request**.**user**.**is\_anonymous**:**

wechat\_id **=** request**.**session**.**get**(**"wechat\_id"**)**

**if** wechat\_id**:**

**try:**

request**.**wechat **=** WechatUserProfile**.**objects**.**get**(**pk**=**wechat\_id**)**

**except:**

**pass**

如果当前WechatUser指定并没有绑定User，则提示绑定

templates.registration.login.html

{% if request.user.is\_anonymous and request.wechat %}

You need to login to continue

{% endif %}

<form method=**"post"** action=**""**>

**{% csrf\_token %}**

**{{ form|crispy }}**

<input class = **"btn btn-block btn-primary"** type=**"submit"** value=**"{% trans 'Log in' %}"** />

**{% if request.user.is\_anonymous and request.wechat %}**

<input type=**"hidden"** name=**"next"** value=**"{% url 'link\_to\_wechat' %}"** /> <!-- workaround here, maybe we need to change it in login function later-->

**{% else %}**

<input type=**"hidden"** name=**"next"** value=**"{{ next }}"** />

**{% endif %}**

</form>

### 解绑WechatUser和User

personalcenter.urls

urlpatterns **=** **[**

url**(**r'^unlinkfromwechat$'**,** account\_unlink\_from\_wechat**,** name**=**'unlink\_from\_wechat'**),**

**def** account\_unlink\_from\_wechat**(**request**):**

wechat\_id **=** request**.**session**.**get**(**"wechat\_id"**)**

wechat **=** **None**

**if** wechat\_id**:**

**try:**

wechat **=** WechatUserProfile**.**objects**.**get**(**pk**=**wechat\_id**)**

wechat**.**user **=** **None**

wechat**.**save**()**

**except:**

**pass**

**return** redirect**(**reverse**(**"home"**,** kwargs**={}))** # to change to ajax

### django实现

#### authenticate

下面分析django是如何实现authentication的

authenticate会直接调用这个函数

django.contrib.auth.\_\_init\_\_

**def** authenticate**(\*\***credentials**):**

"""

If the given credentials are valid, return a User object.

"""

**for** backend**,** backend\_path **in** \_get\_backends**(**return\_tuples**=True):**

**try:**

inspect**.**getcallargs**(**backend**.**authenticate**,** **\*\***credentials**)** # 遍历backend

**except** TypeError**:**

# This backend doesn't accept these credentials as arguments. Try the next one.

**continue**

**try:**

user **=** backend**.**authenticate**(\*\***credentials**)** # 调用backend的authenticate函数

**except** PermissionDenied**:**

# This backend says to stop in our tracks - this user should not be allowed in at all.

**return** **None**

**if** user **is** **None:**

**continue**

# Annotate the user object with the path of the backend.

user**.**backend **=** backend\_path

**return** user

# The credentials supplied are invalid to all backends, fire signal

user\_login\_failed**.**send**(**sender**=**\_\_name\_\_**,**

credentials**=**\_clean\_credentials**(**credentials**))**

demo代码里backend的authenticate的参数包含request，调用authenticate时，必须request=request, 否则django.contrib.auth.\_\_init\_\_的authenticate**(\*\***credentials**)**函数在getcallargs这一步会出错，因为request这个参数不能传进**\*\***credentials

<https://docs.djangoproject.com/en/dev/topics/auth/customizing/#writing-an-authentication-backend>

* [Keyword Arguments](https://docs.python.org/2/tutorial/controlflow.html#keyword-arguments)
* [python 中参数\*args, \*\*kwargs](http://blog.csdn.net/anhuidelinger/article/details/10011013)

网上例子的练习，实际并没有用

*Auth.py (personalcenter\backends)*

class SettingsBackend(object):

**def** authenticate**(**self**,** request**,** username**=None,** password**=None):**

login\_valid **=** **(**settings**.**ADMIN\_LOGIN **==** username**)**

pwd\_valid **=** check\_password**(**password**,** settings**.**ADMIN\_PASSWORD**)**

**if** login\_valid **and** pwd\_valid**:**

**try:**

user **=** User**.**objects**.**get**(**username**=**username**)**

**except** User**.**DoesNotExist**:**

# Create a new user. There's no need to set a password

# because only the password from settings.py is checked.

user **=** User**(**username**=**username**)**

user**.**is\_staff **=** **True**

user**.**is\_superuser **=** **True**

user**.**save**()**

**return** user

**return** **None**

**def** get\_user**(**self**,** user\_id**):**

**try:**

**return** User**.**objects**.**get**(**pk**=**user\_id**)**

**except** User**.**DoesNotExist**:**

**return** **None**

<https://docs.djangoproject.com/en/dev/topics/auth/default/#how-to-log-a-user-in>

**from** django**.**contrib**.**auth **import** authenticate**,** login

**def** my\_view**(**request**):**

username **=** request**.**POST**[**'username'**]**

password **=** request**.**POST**[**'password'**]**

user **=** authenticate**(**request= request**,** username**=**username**,** password**=**password**)**

**if** user **is** **not** **None:**

login**(**request**,** user**)**

# Redirect to a success page.

**...**

**else:**

# Return an 'invalid login' error message.

**...**

*django.contrib.auth.backends.py*

class ModelBackend(object):

"""

Authenticates against settings.AUTH\_USER\_MODEL.

"""

def authenticate(self, username=None, password=None, \*\*kwargs):

UserModel = get\_user\_model()

if username is None:

username = kwargs.get(UserModel.USERNAME\_FIELD)

try:

user = UserModel.\_default\_manager.get\_by\_natural\_key(username)

if user.check\_password(password):

return user

except UserModel.DoesNotExist:

# Run the default password hasher once to reduce the timing

# difference between an existing and a non-existing user (#20760).

UserModel().set\_password(password)

#### login

*\_\_init\_\_.py (site-packages\django\contrib\auth)*

SESSION\_KEY **=** '\_auth\_user\_id'

BACKEND\_SESSION\_KEY **=** '\_auth\_user\_backend'

HASH\_SESSION\_KEY **=** '\_auth\_user\_hash'

REDIRECT\_FIELD\_NAME **=** 'next'

**def** login**(**request**,** user**):**

"""

Persist a user id and a backend in the request. This way a user doesn't

have to reauthenticate on every request. Note that data set during

the anonymous session is retained when the user logs in.

"""

session\_auth\_hash **=** ''

**if** user **is** **None:**

user **=** request**.**user

**if** hasattr**(**user**,** 'get\_session\_auth\_hash'**):**

session\_auth\_hash **=** user**.**get\_session\_auth\_hash**()**

**if** SESSION\_KEY **in** request**.**session**:**

**if** \_get\_user\_session\_key**(**request**)** **!=** user**.**pk **or** **(**

session\_auth\_hash **and**

request**.**session**.**get**(**HASH\_SESSION\_KEY**)** **!=** session\_auth\_hash**):**

# To avoid reusing another user's session, create a new, empty

# session if the existing session corresponds to a different

# authenticated user.

request**.**session**.**flush**()**

**else:**

request**.**session**.**cycle\_key**()**

# 正常用户登录时，会保存这个用户session，这样在response的消息里的request里面会带出去user信息

request**.**session**[**SESSION\_KEY**]** **=** user**.**\_meta**.**pk**.**value\_to\_string**(**user**)**

request**.**session**[**BACKEND\_SESSION\_KEY**]** **=** user**.**backend

request**.**session**[**HASH\_SESSION\_KEY**]** **=** session\_auth\_hash

**if** hasattr**(**request**,** 'user'**):**

request**.**user **=** user

rotate\_token**(**request**)**

user\_logged\_in**.**send**(**sender**=**user**.**\_\_class\_\_**,** request**=**request**,** user**=**user**)**

{{user}}

{{request.user}}

这两个user分别是由下面的middleware传进来的

'django.template.context\_processors.request',

'django.contrib.auth.context\_processors.auth',

#### get\_user

*django/contrib/auth/\_\_init\_\_*

**def** get\_user**(**request**):**

"""

Returns the user model instance associated with the given request session.

If no user is retrieved an instance of `AnonymousUser` is returned.

"""

**from** **.**models **import** AnonymousUser

user **=** **None**

**try:**

user\_id **=** \_get\_user\_session\_key**(**request**)**

backend\_path **=** request**.**session**[**BACKEND\_SESSION\_KEY**]**

**except** KeyError**:**

**pass**

**else:**

**if** backend\_path **in** settings**.**AUTHENTICATION\_BACKENDS**:**

backend **=** load\_backend**(**backend\_path**)**

user **=** backend**.**get\_user**(**user\_id**)**

# Verify the session

**if** **(**'django.contrib.auth.middleware.SessionAuthenticationMiddleware'

**in** settings**.**MIDDLEWARE\_CLASSES **and** hasattr**(**user**,** 'get\_session\_auth\_hash'**)):**

session\_hash **=** request**.**session**.**get**(**HASH\_SESSION\_KEY**)**

session\_hash\_verified **=** session\_hash **and** constant\_time\_compare**(**

session\_hash**,**

user**.**get\_session\_auth\_hash**()**

**)**

**if** **not** session\_hash\_verified**:**

request**.**session**.**flush**()**

user **=** **None**

**return** user **or** AnonymousUser**()**

*django/contrib/auth/backends.py*

class ModelBackend(object):

def get\_user(self, user\_id):

UserModel = get\_user\_model()

try:

return UserModel.\_default\_manager.get(pk=user\_id)

except UserModel.DoesNotExist:

return None

*django/contrib/auth/forms.py*

class AuthenticationForm(forms.Form):

def get\_user(self):

return self.user\_cache

*django/contrib/auth/middleware.py*

**def** get\_user**(**request**):**

**if** **not** hasattr**(**request**,** '\_cached\_user'**):**

request**.**\_cached\_user **=** auth**.**get\_user**(**request**)**

**return** request**.**\_cached\_user

\_cached\_user when will it be used?

#### auth

*django.contrib.auth.context\_processors.auth*

**def** auth**(**request**):**

"""

Returns context variables required by apps that use Django's authentication

system.

If there is no 'user' attribute in the request, uses AnonymousUser (from

django.contrib.auth).

"""

**if** hasattr**(**request**,** 'user'**):**

user **=** request**.**user

**else:**

**from** django**.**contrib**.**auth**.**models **import** AnonymousUser

user **=** AnonymousUser**()**

**return** **{**

'user'**:** user**,**

'perms'**:** PermWrapper**(**user**),**

**}**

#### middleware

*Middleware.py (site-packages\django\contrib\auth*

**def** get\_user**(**request**):**

**if** **not** hasattr**(**request**,** '\_cached\_user'**):**

request**.**\_cached\_user **=** auth**.**get\_user**(**request**)**

**return** request**.**\_cached\_user

**class** **AuthenticationMiddleware(**object**):**

**def** process\_request**(**self**,** request**):**

**assert** hasattr**(**request**,** 'session'**),** **(**

"The Django authentication middleware requires session middleware "

"to be installed. Edit your MIDDLEWARE\_CLASSES setting to insert "

"'django.contrib.sessions.middleware.SessionMiddleware' before "

"'django.contrib.auth.middleware.AuthenticationMiddleware'."

**)**

request**.**user **=** SimpleLazyObject**(lambda:** get\_user**(**request**))**

class ThreadingMixIn:

"""Mix-in class to handle each request in a new thread."""

# Decides how threads will act upon termination of the

# main process

daemon\_threads = False

def process\_request\_thread(self, request, client\_address):

"""Same as in BaseServer but as a thread.

In addition, exception handling is done here.

"""

try:

self.finish\_request(request, client\_address)

self.shutdown\_request(request)

except:

self.handle\_error(request, client\_address)

self.shutdown\_request(request)

def process\_request(self, request, client\_address):

"""Start a new thread to process the request."""

t = threading.Thread(target = self.process\_request\_thread,

args = (request, client\_address))

t.daemon = self.daemon\_threads

t.start()

class BaseHandler(object):

**def** get\_response**(**self**,** request**):**

"Returns an HttpResponse object for the given HttpRequest"

# Setup default url resolver for this thread, this code is outside

# the try/except so we don't get a spurious "unbound local

# variable" exception in the event an exception is raised before

# resolver is set

urlconf **=** settings**.**ROOT\_URLCONF

urlresolvers**.**set\_urlconf**(**urlconf**)**

resolver **=** urlresolvers**.**RegexURLResolver**(**r'^/'**,** urlconf**)**

**try:**

response **=** **None**

# Apply request middleware

**for** middleware\_method **in** self**.**\_request\_middleware**:**

response **=** middleware\_method**(**request**)**

**if** response**:**

**break**

**if** response **is** **None:**

**if** hasattr**(**request**,** 'urlconf'**):**

# Reset url resolver with a custom urlconf.

urlconf **=** request**.**urlconf

urlresolvers**.**set\_urlconf**(**urlconf**)**

resolver **=** urlresolvers**.**RegexURLResolver**(**r'^/'**,** urlconf**)**

resolver\_match **=** resolver**.**resolve**(**request**.**path\_info**)**

callback**,** callback\_args**,** callback\_kwargs **=** resolver\_match

request**.**resolver\_match **=** resolver\_match

# Apply view middleware

**for** middleware\_method **in** self**.**\_view\_middleware**:**

response **=** middleware\_method**(**request**,** callback**,** callback\_args**,** callback\_kwargs**)**

**if** response**:**

**break**

**if** response **is** **None:**

wrapped\_callback **=** self**.**make\_view\_atomic**(**callback**)**

**try:**

response = wrapped\_callback(request, \*callback\_args, \*\*callback\_kwargs) # url对应链接

**except** Exception **as** e**:**

# If the view raised an exception, run it through exception

# middleware, and if the exception middleware returns a

# response, use that. Otherwise, reraise the exception.

**for** middleware\_method **in** self**.**\_exception\_middleware**:**

response **=** middleware\_method**(**request**,** e**)**

**if** response**:**

**break**

**if** response **is** **None:**

**raise**

# Complain if the view returned None (a common error).

**if** response **is** **None:**

**if** isinstance**(**callback**,** types**.**FunctionType**):** # FBV

view\_name **=** callback**.**\_\_name\_\_

**else:** # CBV

view\_name **=** callback**.**\_\_class\_\_**.**\_\_name\_\_ **+** '.\_\_call\_\_'

**raise** ValueError**(**"The view %s.%s didn't return an HttpResponse object. It returned None instead."

**%** **(**callback**.**\_\_module\_\_**,** view\_name**))**

# If the response supports deferred rendering, apply template

# response middleware and then render the response

**if** hasattr**(**response**,** 'render'**)** **and** callable**(**response**.**render**):**

**for** middleware\_method **in** self**.**\_template\_response\_middleware**:**

response **=** middleware\_method**(**request**,** response**)**

# Complain if the template response middleware returned None (a common error).

**if** response **is** **None:**

**raise** ValueError**(**

"%s.process\_template\_response didn't return an "

"HttpResponse object. It returned None instead."

**%** **(**middleware\_method**.**\_\_self\_\_**.**\_\_class\_\_**.**\_\_name\_\_**))**

response **=** response**.**render**()**

**except** http**.**Http404 **as** e**:**

logger**.**warning**(**'Not Found: %s'**,** request**.**path**,**

extra**={**

'status\_code'**:** 404**,**

'request'**:** request

**})**

**if** settings**.**DEBUG**:**

response **=** debug**.**technical\_404\_response**(**request**,** e**)**

**else:**

response **=** self**.**get\_exception\_response**(**request**,** resolver**,** 404**)**

**except** PermissionDenied**:**

logger**.**warning**(**

'Forbidden (Permission denied): %s'**,** request**.**path**,**

extra**={**

'status\_code'**:** 403**,**

'request'**:** request

**})**

response **=** self**.**get\_exception\_response**(**request**,** resolver**,** 403**)**

**except** MultiPartParserError**:**

logger**.**warning**(**

'Bad request (Unable to parse request body): %s'**,** request**.**path**,**

extra**={**

'status\_code'**:** 400**,**

'request'**:** request

**})**

response **=** self**.**get\_exception\_response**(**request**,** resolver**,** 400**)**

**except** SuspiciousOperation **as** e**:**

# The request logger receives events for any problematic request

# The security logger receives events for all SuspiciousOperations

security\_logger **=** logging**.**getLogger**(**'django.security.%s' **%**

e**.**\_\_class\_\_**.**\_\_name\_\_**)**

security\_logger**.**error**(**

force\_text**(**e**),**

extra**={**

'status\_code'**:** 400**,**

'request'**:** request

**})**

**if** settings**.**DEBUG**:**

**return** debug**.**technical\_500\_response**(**request**,** **\***sys**.**exc\_info**(),** status\_code**=**400**)**

response **=** self**.**get\_exception\_response**(**request**,** resolver**,** 400**)**

**except** SystemExit**:**

# Allow sys.exit() to actually exit. See tickets #1023 and #4701

**raise**

**except:** # Handle everything else.

# Get the exception info now, in case another exception is thrown later.

signals**.**got\_request\_exception**.**send**(**sender**=**self**.**\_\_class\_\_**,** request**=**request**)**

response **=** self**.**handle\_uncaught\_exception**(**request**,** resolver**,** sys**.**exc\_info**())**

**try:**

# Apply response middleware, regardless of the response

**for** middleware\_method **in** self**.**\_response\_middleware**:**

response **=** middleware\_method**(**request**,** response**)**

# Complain if the response middleware returned None (a common error).

**if** response **is** **None:**

**raise** ValueError**(**

"%s.process\_response didn't return an "

"HttpResponse object. It returned None instead."

**%** **(**middleware\_method**.**\_\_self\_\_**.**\_\_class\_\_**.**\_\_name\_\_**))**

response **=** self**.**apply\_response\_fixes**(**request**,** response**)**

**except:** # Any exception should be gathered and handled

signals**.**got\_request\_exception**.**send**(**sender**=**self**.**\_\_class\_\_**,** request**=**request**)**

response **=** self**.**handle\_uncaught\_exception**(**request**,** resolver**,** sys**.**exc\_info**())**

response**.**\_closable\_objects**.**append**(**request**)**

**return** response

#### render

render(request, *"home.html"*, context)

*Shortcuts.py (site-packages\django)*

render -> Engine.render\_to\_string -> Template.render -> Context. bind\_template

### 参考

<https://docs.djangoproject.com/en/1.5/ref/settings/#auth-profile-module>

<https://docs.djangoproject.com/en/dev/topics/auth/customizing/#substituting-a-custom-user-model>

[Django中扩展User模型](http://blog.csdn.net/watsy/article/details/15506351)

[非profile方式扩展Django User Model](http://onlypython.group.iteye.com/group/wiki/1519-expansion-django-user-model-by-non-profile-way)

[django user 权限](http://www.cnblogs.com/smallcoderhujin/p/3193103.html)

[django 注册、登录及第三方接口程序(2)：扩展User表](http://www.cnblogs.com/BeginMan/p/3389118.html)

[扩展django的User的部分方法](http://www.cnblogs.com/captain-cp/p/3590213.html)

[django admin框架使用系列之三:扩展user model](http://www.yihaomen.com/article/python/334.htm)

[Django扩展user表并使用email登陆](http://www.python88.com/topic/395/)

[Django 扩展 User model 的字段有什么好的方法？](https://segmentfault.com/q/1010000000610904)

[django使用email进行身份验证](http://guangboo.org/2013/03/27/authentication-using-email-in-django)

[自定义的用户认证如何登录使用Django自带的后台管理?](https://www.oschina.net/question/2341837_235437)

[How to log a user in](https://docs.djangoproject.com/en/dev/topics/auth/default/#how-to-log-a-user-in)

<https://docs.djangoproject.com/en/dev/topics/db/models/#proxy-models>

## 信号Signal

<https://docs.djangoproject.com/en/1.8/ref/signals/#post-save>

以registration库为例：

信号的使用包括以下几个步骤

1. 信号的定义
2. 关联信号和处理函数
3. 发送信号

*Signals.py (site-packages\registration)*

**from** django**.**conf **import** settings

**from** django**.**contrib**.**auth **import** login**,** get\_backends

**from** django**.**dispatch **import** Signal

# A new user has registered.

user\_registered **=** Signal**(**providing\_args**=[**"user"**,** "request"**])**

# A user has activated his or her account.

user\_activated **=** Signal**(**providing\_args**=[**"user"**,** "request"**])** #定义信号

**def** login\_user**(**sender**,** user**,** request**,** **\*\***kwargs**):**

""" Automatically authenticate the user when activated """

backend **=** get\_backends**()[**0**]** # Hack to bypass `authenticate()`.

user**.**backend **=** "%s.%s" **%** **(**backend**.**\_\_module\_\_**,** backend**.**\_\_class\_\_**.**\_\_name\_\_**)**

login**(**request**,** user**)**

request**.**session**[**'REGISTRATION\_AUTO\_LOGIN'**]** **=** **True**

request**.**session**.**modified **=** **True**

**if** getattr**(**settings**,** 'REGISTRATION\_AUTO\_LOGIN'**,** **False):**

user\_activated**.**connect**(**login\_user**)** #关联信号激活事件和处理函数

*Views.py (site-packages\registration)*

class RegistrationView(BaseRegistrationView):

def register(self, form):

"""

Given a username, email address and password, register a new

user account, which will initially be inactive.

Along with the new ``User`` object, a new

``registration.models.RegistrationProfile`` will be created,

tied to that ``User``, containing the activation key which

will be used for this account.

An email will be sent to the supplied email address; this

email should contain an activation link. The email will be

rendered using two templates. See the documentation for

``RegistrationProfile.send\_activation\_email()`` for

information about these templates and the contexts provided to

them.

After the ``User`` and ``RegistrationProfile`` are created and

the activation email is sent, the signal

``registration.signals.user\_registered`` will be sent, with

the new ``User`` as the keyword argument ``user`` and the

class of this backend as the sender.

"""

site = get\_current\_site(self.request)

if hasattr(form, 'save'):

new\_user\_instance = form.save()

else:

new\_user\_instance = (UserModel().objects

.create\_user(\*\*form.cleaned\_data))

new\_user = RegistrationProfile.objects.create\_inactive\_user(

new\_user=new\_user\_instance,

site=site,

send\_email=self.SEND\_ACTIVATION\_EMAIL,

request=self.request,

)

signals.user\_registered.send(sender=self.\_\_class\_\_,

user=new\_user,

request=self.request)

return new\_user

class ActivationView(BaseActivationView):

def activate(self, \*args, \*\*kwargs):

"""

Given an an activation key, look up and activate the user

account corresponding to that key (if possible).

After successful activation, the signal

``registration.signals.user\_activated`` will be sent, with the

newly activated ``User`` as the keyword argument ``user`` and

the class of this backend as the sender.

"""

activation\_key = kwargs.get('activation\_key', '')

activated\_user = (RegistrationProfile.objects

.activate\_user(activation\_key))

if activated\_user:

signals.user\_activated.send(sender=self.\_\_class\_\_,

user=activated\_user,

request=self.request)

return activated\_user

def get\_success\_url(self, user):

return ('registration\_activation\_complete', (), {})

参考文档

* [django拾遗之signal](http://www.cnblogs.com/mindsbook/archive/2009/10/27/django_signal.html)

## Search Query

028 Distinct Random QuerySets

## Formset

019 Formset for Inventory

## LoginRequired

020 Login Required Mixins

## Django Message

## Django Session

040 Django sessions

## 轮播图

<http://getbootstrap.com/javascript/#carousel>

## 评论

django.contrib.comments在django1.8已删除

[ImportError: No module named comments django](http://stackoverflow.com/questions/32986302/importerror-no-module-named-comments-django)

Django comments are deprecated in latest version. You can either install older django version (such as 1.5). Or the better solution is to install comments from external repository like so:

pip install django-contrib-comments

and then change all imports like this:

**from** django**.**contrib**.**comments**.**models **import** Comment # old

**from** django\_comments**.**models **import** Comment # new

Django1.9版本使用django\_comments而非“django.contrib.comments”，直接pip install django-contrib-comments即可。

可选的库包括

* mptt
* threadedcomments

最后选用mptt

生成的数据库效果图





**object\_pk**

Mptt第一级的object\_pk是crowdfunding，但是子评论的object\_pk是父评论

注: 9是手动改的level变成1，变成8的子，用些测试目的。所以object\_id=2，因为创建是level为0

评论显示的树形结构

|  |  |  |  |
| --- | --- | --- | --- |
| 8 |  |  |  |
|  | 9 |  |  |
|  |  | 12 |  |
|  |  | 13 |  |
|  |  |  | 20 |
|  | 11 |  |  |
|  | 14 |  |  |
| 10 |  |  |  |
|  | 15 |  |  |
|  | 16 |  |  |
|  | 17 |  |  |
|  | 19 |  |  |
| 18 |  |  |  |

html结构如下



[django-mptt](https://github.com/django-mptt/django-mptt/) (github)

<http://django-mptt.readthedocs.io/en/latest/index.html>

<http://django-mptt.readthedocs.io/en/latest/tutorial.html#getting-started>

mfitzp.io

[Django AJAX threaded-comments using only jQuery](http://mfitzp.io/article/django-ajax-threaded-comments-using-only-jquery/)

[Django Threaded Comments with mptt](http://mfitzp.io/article/django-threaded-comments/" \o "Django Threaded Comments)

[Syntax highlighting with Django and Markdown](http://mfitzp.io/article/syntax-highlighting-with-django-and-markdown/)

slog

[django 简易博客开发 4 comments库使用及ajax支持](http://www.cnblogs.com/cacique/archive/2012/10/03/2710803.html)

[Django mptt介绍以及使用](http://qinxuye.me/article/introduction-and-usage-of-django-mptt/)

django-threadedcomments

<https://pypi.python.org/pypi/django-threadedcomments>

<https://github.com/HonzaKral/django-threadedcomments>

[Eric Florenzano’s Blog - Category: django-threadedcomments](http://eflorenzano.com/blog/categories/django-threadedcomments/)

django-comments-threaded

<https://github.com/marazmiki/django-comments-threaded>

django-contrib-comments / django\_comments

<http://django-comments-xtd.readthedocs.io/en/latest/>

[Django 自带comment评论库使用](http://www.cnblogs.com/weiming-cheng/p/5344309.html)

[Django个人博客开发（五）](https://www.dannysite.com/blog/16/)

djangocms-comments

[https://pypi.python.org/pypi/](https://pypi.python.org/pypi/djangocms-comments/0.2.2)[djangocms-comments/0.2.2](https://pypi.python.org/pypi/djangocms-comments/0.2.2)

[django中的多级评论](http://www.cnblogs.com/ccorz/p/5985534.html)

[Django中使用jQuery做评论的无刷新提交](http://www.cnblogs.com/restran/archive/2012/04/26/2469839.html)

[Django使用Ajax实现页面无刷新评论回复功能](http://www.cnblogs.com/mfc-itblog/p/5188900.html)

[分层评论的实现思路,thread comments 实现思路,数据库存树状结构](http://www.yihaomen.com/article/python/255.htm)

社会化评论

[django-blog-zinnia集成多说评论框](http://bookshadow.com/weblog/2014/06/04/django-blog-zinnia-add-duoshuo/)

[Disqus\_百度百科](https://www.baidu.com/link?url=U33F7R5Dlgc7qtGN0njPDD9_gO-Dzn3xf1zgeVeIR4S4wLpr_pt4N03camgNEW4l50SFSUEaHZSTIfxcr6VxD_&wd=&eqid=d9d7b7a80003b0790000000458440bd3)

[国内社会化评论(类DISQUS)插件哪个好?](https://www.baidu.com/link?url=VPTczqkq7L-HDLvGHU7UfJ4PTkb8CNzKMlJitI7kyskHi9TXfI9npoopHK06W4Cy&wd=&eqid=e510e6f10003aa960000000458440c69)

### sblog

[sblog](https://github.com/gusibi/sblog)怎么做comments的ajax更新的

*templates/simpleblog/blog\_add\_comments.html*

<div id="comment\_form">

<form class="form-horizontal" action="{% comment\_form\_target %}" method="post">

<fieldset>

<div class="form-actions">

<input class="btn btn-info" type="submit" name="submit" value="Post">

{# <input class="btn btn-info" type="submit" name="preview" value="Preview"> #}

<input type='hidden' name='next' value='/simpleblog/blog/{{ blog.id }}/commentshow/'/>

</div>

</fieldset>

</form>

</div>

*simpleblog/urls.py*

url**(**r'^blog/(?P<id>\d+)/commentshow/$'**,** 'blog\_show\_comment'**,** name**=**'showcomment'**),**

*simpleblog/views.py*

**def** blog\_show\_comment**(**request**,** id**=**''**):**

blog **=** Blog**.**objects**.**get**(**id**=**id**)**

**return** render\_to\_response**(**'blog\_comments\_show.html'**,** **{**"blog"**:** blog**})**

*templates/simpleblog/blog\_comments\_show.html*

{% load comments %}

{% load gravatar %}

{% load markup %}

<article id="cmt">

{% get\_comment\_count for blog as comment\_count %}

<h4 class="muted">{{ comment\_count }} Comments</h4>

<hr class="soften">

{% get\_comment\_list for blog as blog\_com %}

{% for comment in blog\_com %}

<div class="container-fluid none-padding">

<div class="row-fluid">

<div class="span1">

<img class="gravatar" src="{% gravatar\_url comment.user\_email %}">

</div>

<div class="span10">

<p class="muted"><small>{{ comment.user }}</small><small>{{ comment.submit\_date|date:"F,j,Y" }}</small></p>

{{ comment.comment|markdown:'safe,codehilite' }}

</div>

</div>

</div>

<hr class="soften">

{% endfor %}

</article>

*templates/simpleblog/blog\_show.html*

<script type="text/javascript" charset="utf-8">

function bindPostCommentHandler() {

$('#comment\_form form input.submit-preview').remove();

$('#comment\_form form').submit(function() {

$.ajax({

type: "POST",

data: $('#comment\_form form').serialize(),

url: "{% comment\_form\_target %}",

cache: false,

dataType: "html",

success: function(html, textStatus) {

$('#cmt').replaceWith(html);

$('#comment\_form form')[0].reset();

},

error: function (XMLHttpRequest, textStatus, errorThrown) {

$('#comment\_form form').replaceWith('Your comment was unable to be posted at this time. We apologise for the inconvenience.');

}

});

return false;

});

}

$(document).ready(function() {

bindPostCommentHandler();

});

</script>

**def** post\_comment**(**request**,** next**=None,** using**=None):**

我的方法

1. Form初始化为hide，点击回复显示
2. 整个comments替换
3. 替换后 整体ajax要重新加载

网上方法

1. Form动态添加，怎么解决node问题？
2. 只替换部分comment，form改成comment，form的位置可能要改
3. 只需要加载新增加部分的ajax

### CSS

[HTML 5 <article> 标签](http://www.w3school.com.cn/html5/html5_article.asp)

[Bootstrap Well](http://www.runoob.com/bootstrap/bootstrap-wells.html)

### Issue

Reverse for 'comments-post-comment' with arguments '()' and keyword arguments '{}' not found. 1 pattern(s) tried: ['crowdfundings/comments$post/$']

url**(**r'^comments$'**,** include**(**'django\_comments.urls'**)),**

url**(**r'^comments/'**,** include**(**'django\_comments.urls'**)),**

### Jquery

[jquery after append appendTo三个函数的区别](http://blog.csdn.net/ljyabc1028/article/details/7341665)

[jQuery选择器总结](http://www.cnblogs.com/onlys/articles/jQuery.html)

[jQuery 参考手册 - 遍历](http://www.w3school.com.cn/jquery/jquery_ref_traversing.asp)

[jquery选择器 之 获取父级元素、同级元素、子元素](http://blog.163.com/zhaoyanping_1125/blog/static/2013291532011112961637321/)

### templatetag

[django中自定义标签和过滤器](http://www.cnblogs.com/MnCu8261/p/5934203.html)

## 分页

[Django 分页插件：django-pagination](http://blog.csdn.net/l_courser/article/details/7038859)

## 跳转

登录成功，密码修改成功之后，跳转回原来的网址

以admin的auth为例：

在template里面定义好’next’网址，并通过GET/POST方法传递给后台，后台在login成功之后会重定向到该网址

有时需要获取当前site网址，可以使用get\_current\_site方法

*\_\_init\_\_.py (site-packages\django\contrib\auth)*

REDIRECT\_FIELD\_NAME **=** 'next'

*Views.py (site-packages\django\contrib\auth)*

@sensitive\_post\_parameters()

@csrf\_protect

@never\_cache

def login(request, template\_name='registration/login.html',

redirect\_field\_name=REDIRECT\_FIELD\_NAME,

authentication\_form=AuthenticationForm,

current\_app=None, extra\_context=None):

"""

Displays the login form and handles the login action.

"""

redirect\_to = request.POST.get(redirect\_field\_name,

request.GET.get(redirect\_field\_name, ''))

if request.method == "POST":

form = authentication\_form(request, data=request.POST)

if form.is\_valid():

# Ensure the user-originating redirection url is safe.

if not is\_safe\_url(url=redirect\_to, host=request.get\_host()):

redirect\_to = resolve\_url(settings.LOGIN\_REDIRECT\_URL)

# Okay, security check complete. Log the user in.

auth\_login(request, form.get\_user())

return HttpResponseRedirect(redirect\_to)

else:

form = authentication\_form(request)

current\_site = get\_current\_site(request)

context = {

'form': form,

redirect\_field\_name: redirect\_to,

'site': current\_site,

'site\_name': current\_site.name,

}

if extra\_context is not None:

context.update(extra\_context)

if current\_app is not None:

request.current\_app = current\_app

return TemplateResponse(request, template\_name, context)

*Shortcuts.py (site-packages\django\contrib\sites)*

**def** get\_current\_site**(**request**):**

"""

Checks if contrib.sites is installed and returns either the current

``Site`` object or a ``RequestSite`` object based on the request.

"""

# Imports are inside the function because its point is to avoid importing

# the Site models when django.contrib.sites isn't installed.

**if** apps**.**is\_installed**(**'django.contrib.sites'**):**

**from** **.**models **import** Site

**return** Site**.**objects**.**get\_current**(**request**)**

**else:**

**from** **.**requests **import** RequestSite

**return** RequestSite**(**request**)**

*Models.py (site-packages\django\contrib\sites)*

class SiteManager(models.Manager):

def get\_current(self, request=None):

"""

Returns the current Site based on the SITE\_ID in the project's settings.

If SITE\_ID isn't defined, it returns the site with domain matching

request.get\_host(). The ``Site`` object is cached the first time it's

retrieved from the database.

"""

from django.conf import settings

if getattr(settings, 'SITE\_ID', ''):

site\_id = settings.SITE\_ID

return self.\_get\_site\_by\_id(site\_id)

elif request:

return self.\_get\_site\_by\_request(request)

raise ImproperlyConfigured(

"You're using the Django \"sites framework\" without having "

"set the SITE\_ID setting. Create a site in your database and "

"set the SITE\_ID setting or pass a request to "

"Site.objects.get\_current() to fix this error."

)

举例：

*Login.html (site-packages\registration\templates\registration)*

<form method=**"post"** action=**""**>

**{% csrf\_token %}**

**{{ form.as\_p }}**

<input type=**"submit"** value=**"{% trans 'Log in' %}"** />

<input type=**"hidden"** name=**"next"** value=**"{{ next }}"** />

</form>

该next值会传递给login函数

## Admin 扩展

评论模块

django-grapplli，它主要就是修改了django自动后台的界面，另外就是django-filebrowser，它其实是和grapplli集成的，主要用来在后台管理静态文件。

[django grappelli](http://www.cnblogs.com/freeggache/p/4450637.html)

[用grappelli美化Django Admin后台管理界面](http://blog.csdn.net/huyoo/article/details/6965816)

[在Django中使用markdown](http://blog.csdn.net/mingzznet/article/details/50688192)

django 简易博客开发 1 安装、创建、配置、admin使用   
<http://www.cnblogs.com/cacique/archive/2012/09/29/2707976.html>   
django 简易博客开发 2 模板和数据查询   
<http://www.cnblogs.com/cacique/archive/2012/09/30/2709143.html>   
django 简易博客开发 3 静态文件、from 应用与自定义   
<http://www.cnblogs.com/cacique/archive/2012/10/01/2709668.html>   
django 简易博客开发 4 comments库使用及ajax支持   
<http://www.cnblogs.com/cacique/archive/2012/10/03/2710803.html>   
django 简易博客开发 5 markdown支持、代码高亮、gravatar头像服务   
<http://www.cnblogs.com/cacique/archive/2012/10/07/2713703.html>

035 Product Editing with django Inline

## Social Share

### 微信分享

[微信公众平台开发(107) 分享到朋友圈和发送给好友](http://www.cnblogs.com/txw1958/p/weixin-js-sharetimeline.html)

[Python微信-- 分享接口（分享到朋友圈、朋友、空间）](http://www.cnblogs.com/wasayezi/p/5275691.html)

[微信JS-SDK 分享到朋友圈 分享给朋友 分享到QQ 拍照或从手机相册中选图 识别音频并返回识别结果 使用微信内置地图查看位置](http://www.aichengxu.com/view/5250400)

[微信朋友圈分享页面（JS-SDK 1.0）](http://www.aichengxu.com/view/4578913)

[微信js sdk invalid signature签名错误 问题解决](https://my.oschina.net/u/2308739/blog/371414)

[微信公众平台开发(90) 分享到朋友圈和发送给好友](http://www.cnblogs.com/txw1958/p/MicroMessenger-browser.html)

<http://jsbin.com/kalenahana/edit?html,js,output>

[微信WeixinJSBridge API](http://www.cnblogs.com/txw1958/p/WeixinJSBridge-api.html)

## JQuery

033 Shopping cart icon

088 Cart Count in Navbar

## Ajax