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

Zakkabag

## 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'**,**

'personalcenter.backends.auth.MyBackend'**,**

'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

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

### 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

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)

## 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>

## 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)