django-accounts

Release 1.0

Dec 02, 2020

Contents:

CHAPTER 1

Accounts

1.1 Authentication

class accounts.authentication.**EmailAuthBackend**
Bases: object

Authenticate using e-mail address.

authenticate(*requset*, *username=None*, *password=None*)

get_user(*user_id*)

class accounts.authentication.**PhoneAuthBackend**
Bases: object

Authenticate using phone number

authenticate(*requset*, *username=None*, *password=None*)

get_user(*user_id*)

class accounts.authentication.**UsernameAuthBackend**
Bases: object

Overriding UserBackend it doesn't work on before email activation

authenticate(*requset*, *username=None*, *password=None*)

get_user(*user_id*)

1.2 Device Generator

accounts.device_generator.**get_ip**(*request*)
Returns the IP of the request, accounting for the possibility of being behind a proxy.

accounts.device_generator.**get_location**(*request*)

accounts.device_generator.**get_user_agent**(*request*)

1.3 Forms

class accounts.forms.**LoginForm**(**args*, ***kwargs*)
Bases: django.forms.forms.Form

base_fields = {'password': <django.forms.fields.CharField object>, 'username': <djan

clean(***kwargs*)
Hook for doing any extra form-wide cleaning after Field.clean() has been called on every field. Any ValidationError raised by this method will not be associated with a particular field; it will have a special-case association with the field named '__all__'.

declared_fields = {'password': <django.forms.fields.CharField object>, 'username': <

login(*request*)

media

class accounts.forms.**PhoneVerificationForm**(*data=None*, *files=None*, *auto_id='id_%s'*, *prefix=None*, *initial=None*, *error_class=<class 'django.forms.utils.ErrorList'>*, *label_suffix=None*, *empty_permitted=False*, *instance=None*, *use_required_attribute=None*, *renderer=None*)
Bases: django.forms.models.ModelForm

class Meta
Bases: object

fields = ('phone',)

model
alias of *accounts.models.Profile*

base_fields = {'phone': <phonenumber_field.formfields.PhoneNumberField object>}

clean_phone()

declared_fields = {'phone': <phonenumber_field.formfields.PhoneNumberField object>}

media

class accounts.forms.**ProfileEditForm**(*data=None*, *files=None*, *auto_id='id_%s'*, *prefix=None*, *initial=None*, *error_class=<class 'django.forms.utils.ErrorList'>*, *label_suffix=None*, *empty_permitted=False*, *instance=None*, *use_required_attribute=None*, *renderer=None*)
Bases: django.forms.models.ModelForm

class Meta
Bases: object

fields = ('date_of_birth', 'photo')

model
alias of *accounts.models.Profile*

widgets = {'date_of_birth': <django.forms.widgets.DateInput object>}

base_fields = {'date_of_birth': <django.forms.fields.DateField object>, 'photo': <dj

declared_fields = {}

media

class accounts.forms.**TokenVerificationForm**(*request*, **args*, ***kwargs*)
Bases: django.forms.forms.Form

base_fields = {'token': <django.forms.fields.CharField object>}

declared_fields = {'token': <django.forms.fields.CharField object>}

is_valid()
Return True if the form has no errors, or False otherwise.

media

class accounts.forms.**TrustedDeviceForm**(*data=None*, *files=None*, *auto_id='id_%s'*, *prefix=None*, *initial=None*, *error_class=<class 'django.forms.utils.ErrorList'>*, *label_suffix=None*, *empty_permitted=False*, *instance=None*, *use_required_attribute=None*, *renderer=None*)
Bases: django.forms.models.ModelForm

class Meta
Bases: object

fields = ['trusted']

model
alias of *accounts.models.Device*

widgets = {'trusted': <django.forms.widgets.HiddenInput object>}

base_fields = {'trusted': <django.forms.fields.BooleanField object>}

declared_fields = {}

media

class accounts.forms.**UserEditForm**(*data=None*, *files=None*, *auto_id='id_%s'*, *prefix=None*, *initial=None*, *error_class=<class 'django.forms.utils.ErrorList'>*, *label_suffix=None*, *empty_permitted=False*, *instance=None*, *use_required_attribute=None*, *renderer=None*)
Bases: django.forms.models.ModelForm

class Meta
Bases: object

fields = ('username', 'first_name', 'last_name', 'email', 'phone')

model
alias of *accounts.models.User*

base_fields = {'email': <django.forms.fields.EmailField object>, 'first_name': <djan

declared_fields = {}

media

class accounts.forms.**UserRegistrationForm**(*data=None*, *files=None*, *auto_id='id_%s'*, *prefix=None*, *initial=None*, *error_class=<class 'django.forms.utils.ErrorList'>*, *label_suffix=None*, *empty_permitted=False*, *instance=None*, *use_required_attribute=None*, *renderer=None*)

Bases: django.forms.models.ModelForm

class Meta

Bases: object

fields = ('username', 'phone', 'email')

model

alias of *accounts.models.User*

base_fields = {'email': <django.forms.fields.EmailField object>, 'password': <django

clean_password2()

declared_fields = {'password': <django.forms.fields.CharField object>, 'password2':

media

1.4 Models

class accounts.models.**Activity**(*id*, *user*, *verb*, *target_ct*, *target_id*, *created*)

exception DoesNotExist

exception MultipleObjectsReturned

created

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

get_next_by_created(*, *field=<django.db.models.fields.DateTimeField: created>*, *is_next=True*, ***kwargs*)

get_previous_by_created(*, *field=<django.db.models.fields.DateTimeField: created>*, *is_next=False*, ***kwargs*)

id

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

objects = <django.db.models.manager.Manager object>

target

Provide a generic many-to-one relation through the content_type and object_id fields.

This class also doubles as an accessor to the related object (similar to ForwardManyToOneDescriptor) by adding itself as a model attribute.

target_ct

Accessor to the related object on the forward side of a many-to-one or one-to-one (via ForwardOneToOneDescriptor subclass) relation.

In the example:

```
class Child(Model):
    parent = ForeignKey(Parent, related_name='children')
```

`Child.parent` is a `ForwardManyToOneDescriptor` instance.

target_ct_id

target_id
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

user
Accessor to the related object on the forward side of a many-to-one or one-to-one (via ForwardOneToOneDescriptor subclass) relation.

In the example:

```
class Child(Model):
    parent = ForeignKey(Parent, related_name='children')
```

`Child.parent` is a `ForwardManyToOneDescriptor` instance.

user_id

verb
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

class accounts.models.**Device**(*id*, *user*, *machine*, *browser*, *operating_system*, *ip*, *location*, *trusted*, *created*)

exception DoesNotExist

exception MultipleObjectsReturned

browser
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

check_device_signature(*request*, *user*)
Keep record of user's Host and machine Mac_address

created
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

generate_new_signature(*request*)

get_absolute_url()

get_next_by_created(*, *field=<django.db.models.fields.DateTimeField: created>*, *is_next=True*, ***kwargs*)

get_previous_by_created(*, *field=<django.db.models.fields.DateTimeField: created>*, *is_next=False*, ***kwargs*)

id
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

ip
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

location
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

machine
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

static notify_user(*request*, *device*)

objects = <django.db.models.manager.Manager object>

operating_system
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

trusted
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

user
Accessor to the related object on the forward side of a many-to-one or one-to-one (via ForwardOneToOneDescriptor subclass) relation.

In the example:

```
class Child(Model):
    parent = ForeignKey(Parent, related_name='children')
```

`Child.parent` is a `ForwardManyToOneDescriptor` instance.

user_id

class accounts.models.**Profile**(*id*, *user*, *date_of_birth*, *photo*, *email_verified*, *phone_verified*, *temp_token*)

exception DoesNotExist

exception MultipleObjectsReturned

date_of_birth
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

email_verified
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

id
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

objects = <django.db.models.manager.Manager object>

phone_verified
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

photo
Just like the FileDescriptor, but for ImageFields. The only difference is assigning the width/height to the width_field/height_field, if appropriate.

temp_token
A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

user
Accessor to the related object on the forward side of a one-to-one relation.

In the example:

```
class Restaurant(Model):
    place = OneToOneField(Place, related_name='restaurant')
```

`Restaurant.place` is a `ForwardOneToOneDescriptor` instance.

user_id

class accounts.models.**User**(*id*, *password*, *last_login*, *is_superuser*, *username*, *first_name*, *last_name*, *email*, *is_staff*, *is_active*, *date_joined*, *phone*)

exception DoesNotExist

exception MultipleObjectsReturned

actions
Accessor to the related objects manager on the reverse side of a many-to-one relation.

In the example:

```
class Child(Model):
    parent = ForeignKey(Parent, related_name='children')
```

`Parent.children` is a `ReverseManyToOneDescriptor` instance.

Most of the implementation is delegated to a dynamically defined manager class built by `create_forward_many_to_many_manager()` defined below.

check_verified_fields()

device_set
Accessor to the related objects manager on the reverse side of a many-to-one relation.

In the example:

```
class Child(Model):
    parent = ForeignKey(Parent, related_name='children')
```

`Parent.children` is a `ReverseManyToOneDescriptor` instance.

Most of the implementation is delegated to a dynamically defined manager class built by `create_forward_many_to_many_manager()` defined below.

get_absolute_url()

get_next_by_date_joined(*, *field=<django.db.models.fields.DateTimeField: date_joined>*, *is_next=True*, ***kwargs*)

get_previous_by_date_joined(*, *field=<django.db.models.fields.DateTimeField: date_joined>*, *is_next=False*, ***kwargs*)

groups
Accessor to the related objects manager on the forward and reverse sides of a many-to-many relation.

In the example:

```
class Pizza(Model):
    toppings = ManyToManyField(Topping, related_name='pizzas')
```

`Pizza.toppings` and `Topping.pizzas` are `ManyToManyDescriptor` instances.

Most of the implementation is delegated to a dynamically defined manager class built by `create_forward_many_to_many_manager()` defined below.

id

A wrapper for a deferred-loading field. When the value is read from this object the first time, the query is executed.

logentry_set

Accessor to the related objects manager on the reverse side of a many-to-one relation.

In the example:

```
class Child(Model):
    parent = ForeignKey(Parent, related_name='children')
```

`Parent.children` is a `ReverseManyToOneDescriptor` instance.

Most of the implementation is delegated to a dynamically defined manager class built by `create_forward_many_to_many_manager()` defined below.

phone

The descriptor for the phone number attribute on the model instance. Returns a PhoneNumber when accessed so you can do stuff like:

```
>>> instance.phone_number.as_international
```

Assigns a phone number object on assignment so you can do:

```
>>> instance.phone_number = PhoneNumber(...)
```

or,

```
>>> instance.phone_number = '+414204242'
```

profile

Accessor to the related object on the reverse side of a one-to-one relation.

In the example:

```
class Restaurant(Model):
    place = OneToOneField(Place, related_name='restaurant')
```

`Place.restaurant` is a `ReverseOneToOneDescriptor` instance.

save(**args*, ***kwargs*)

Save the current instance. Override this in a subclass if you want to control the saving process.

The 'force_insert' and 'force_update' parameters can be used to insist that the "save" must be an SQL insert or update (or equivalent for non-SQL backends), respectively. Normally, they should not be set.

user_permissions

Accessor to the related objects manager on the forward and reverse sides of a many-to-many relation.

In the example:

```
class Pizza(Model):
    toppings = ManyToManyField(Topping, related_name='pizzas')
```

`Pizza.toppings` and `Topping.pizzas` are `ManyToManyDescriptor` instances.

Most of the implementation is delegated to a dynamically defined manager class built by `create_forward_many_to_many_manager()` defined below.

1.5 Signals

accounts.signals.**create_user_profile**(*sender*, *instance*, *created*, ***kwargs*)

1.6 Tokens

class accounts.tokens.**AccountActivationTokenGenerator**

Bases: `django.contrib.auth.tokens.PasswordResetTokenGenerator`

Generate account activation token from user_id and timestamp and email verified integer using six lib

1.7 Urls

accounts.urls.**path**(*route*, *view*, *kwargs=None*, *name=None*, ***, *Pattern=<class 'django.urls.resolvers.RoutePattern'>*)

accounts.urls.**re_path**(*route*, *view*, *kwargs=None*, *name=None*, ***, *Pattern=<class 'django.urls.resolvers.RegexPattern'>*)

1.8 Verification

class accounts.verification.**Verificator**(*request*, **args*, ***kwargs*)

Bases: `object`

check_token()

fetch_user_phone_from_session()

generate_token()

process_verification_request(*user*)

send_verification_sms()

verify()

1.9 Views

class accounts.views.**LoginView**(***kwargs*)

Bases: `django.views.generic.base.View`

form

alias of *accounts.forms.LoginForm*

get(*request*, **args*, ***kwargs*)

post(*request*, **args*, ***kwargs*)

template_name = 'registration/login.html'

class accounts.views.**ProfileEditView**(***kwargs*)
Bases: django.contrib.auth.mixins.LoginRequiredMixin, django.views.generic.base.View

get(*request*, **args*, ***kwargs*)

post(*request*, **args*, ***kwargs*)

profile_form
alias of *accounts.forms.ProfileEditForm*

template_name = 'accounts/edit.html'

user_form
alias of *accounts.forms.UserEditForm*

class accounts.views.**RegisterView**(***kwargs*)
Bases: django.views.generic.base.View

form
alias of *accounts.forms.UserRegistrationForm*

get(*request*, **args*, ***kwargs*)

post(*request*, **args*, ***kwargs*)

template_name = 'accounts/register.html'

accounts.views.**activate**(*request*, *uidb64*, *token*)

accounts.views.**activate_phone**(*request*)

accounts.views.**alert_user**(*request*)

accounts.views.**profile**(*request*)

accounts.views.**request_verification_email**(*request*, ***kwargs*)

accounts.views.**user_detail**(*request*, *username*)

accounts.views.**verify_phone**(*request*)

CHAPTER 2

Miscellaneous

2.1 Activity Recorder Mixin

class accounts.activity_recorder_mixin.**RecordsActivityMixin**

Bases: object

target_model: must be an instance of a model with _meta attribute you can pass it as kwargs

static create_action(*user*, *verb*, *target=None*)

dispatch(*request*, **args*, ***kwargs*)

2.2 Decorators

accounts.decorators.**auth_guest**(*function=None*, *redirect_field_name='next'*, *login_url=None*)

Decorator for views that checks that the user is guest in, redirecting to the profile page if necessary.

2.3 Utils

class accounts.utils.**CustomImageField**(*verbose_name=None*, *name=None*, *width_field=None*, *height_field=None*, ***kwargs*)

Bases: django.db.models.fields.files.ImageField

save_form_data(*instance*, *data*)

accounts.utils.**assign_permissions**(*grp_name: str = 'owner_admin'*, *ltd_access_apps: list = []*, *full_access_apps: list = []*, *restricted_models: list = []*)

shortcut to assign permissions

Parameters

- **grp_name** – string of a group name example: 'owner_admin'.

- **ltd_access_apps** – a list of dictionary [{'app_name':['actions']}] or [{'app_name.model':['add','change','delete']}]

  example:[{'accounts': ['delete']},] Note: in case used dotted notation for the same app_name then you have to assign it all with app_name.model style otherwise the app_name only will take precedence.

- **full_access_apps** – list ['app_name'] example: ['accounts', 'store']
- **restricted_models** – list of dotted notation ['app_name.model'] or even ['app_name.model.permission'] example: ['accounts.device'] to remove any ltd_access or full_access model level permissions

- examples:

```
assign_permissions(ltd_access=[{'accounts':['delete']}], full_access=[
↪'store'],
restricted_models=['accounts.activity'])

assign_permissions(
ltd_access=[{'accounts':['delete']}, {'clients':['delete','add','change
↪']}],
full_access=['store'],
restricted_models=['accounts.activity.add','accounts.activity.delete']
)
```

accounts.utils.**create_action**(*user*, *verb*, *target=None*)

helper function to create activity

Parameters

- **user** – user object
- **verb** – string [create, change, delete, add]
- **target** – target model string defaults to None

accounts.utils.**send_verification_email**(*request*, *user*)

2.4 Admin

class accounts.admin.**ActivityAdmin**(*model*, *admin_site*)

Bases: django.contrib.admin.options.ModelAdmin

list_display = ('user', 'verb', 'target', 'created')

list_filter = ('created',)

media

model

alias of *accounts.models.Activity*

search_fields = ('verb',)

class accounts.admin.**CustomUserAdmin**(*model*, *admin_site*)

Bases: django.contrib.auth.admin.UserAdmin

actions = ['suspend_account', 'activate_account']

activate_account(*request*, *queryset*)

email_verified(*obj*)

fieldsets = ((None, {'fields': ('username', 'password')}), ('Personal info', {'fields

inlines = (<class 'accounts.admin.ProfileAdmin'>, <class 'accounts.admin.DeviceInlineA

static last_login_at(*obj*)

static last_login_location(*obj*)

list_display = ['username', 'email', 'email_verified', 'phone', 'phone_verified', 'is_

list_filter = ('is_active', 'profile__email_verified')

list_select_related = ['profile']

media

phone_verified(*obj*)

suspend_account(*request*, *queryset*)

class accounts.admin.**DeviceAdmin**(*model*, *admin_site*)
Bases: django.contrib.admin.options.ModelAdmin, django.contrib.admin.filters.RelatedFieldListFilter

list_display = ('user', 'ip', 'machine', 'location', 'operating_system')

list_filter = ('user', 'machine', 'location')

media

model
alias of *accounts.models.Device*

class accounts.admin.**DeviceInlineAdmin**(*parent_model*, *admin_site*)
Bases: django.contrib.admin.options.TabularInline

extra = 0

media

model
alias of *accounts.models.Device*

verbose_name = 'Active Device'

verbose_name_plural = 'Active Devices'

class accounts.admin.**LogEntryAdmin**(*model*, *admin_site*)
Bases: django.contrib.admin.options.ModelAdmin

date_hierarchy = 'action_time'

has_add_permission(*request*)
Return True if the given request has permission to add an object. Can be overridden by the user in subclasses.

has_change_permission(*request*, *obj=None*)
Return True if the given request has permission to change the given Django model instance, the default implementation doesn't examine the *obj* parameter.

Can be overridden by the user in subclasses. In such case it should return True if the given request has permission to change the *obj* model instance. If *obj* is None, this should return True if the given request has permission to change *any* object of the given type.

has_delete_permission(*request*, *obj=None*)
Return True if the given request has permission to change the given Django model instance, the default implementation doesn't examine the *obj* parameter.

Can be overridden by the user in subclasses. In such case it should return True if the given request has permission to delete the *obj* model instance. If *obj* is None, this should return True if the given request has permission to delete *any* object of the given type.

has_view_permission(*request*, *obj=None*)
Return True if the given request has permission to view the given Django model instance. The default implementation doesn't examine the *obj* parameter.

If overridden by the user in subclasses, it should return True if the given request has permission to view the *obj* model instance. If *obj* is None, it should return True if the request has permission to view any object of the given type.

list_display = ['action_time', 'user', 'content_type', 'object_link', 'action_flag']

list_filter = ['user', 'content_type', 'action_flag']

media

object_link(*obj*)

search_fields = ['object_repr', 'change_message']

class accounts.admin.**ProfileAdmin**(*parent_model*, *admin_site*)
Bases: django.contrib.admin.options.TabularInline

media

model
alias of *accounts.models.Profile*

readonly_fields = ('temp_token',)

CHAPTER 3

Indices and tables

- genindex
- modindex
- search

Python Module Index

a

Index

D

E

F

G

H

I

L

M

N

O

P

R

S

T

U

V

W