
django-accounts

Release 1.0

Dec 02, 2020

Contents:

1	Accounts	1
1.1	Authentication	1
1.2	Device Generator	1
1.3	Forms	2
1.4	Models	4
1.5	Signals	9
1.6	Tokens	9
1.7	Urls	9
1.8	Verification	9
1.9	Views	9
2	Miscellaneous	11
2.1	Activity Recorder Mixin	11
2.2	Decorators	11
2.3	Utils	11
2.4	Admin	12
3	Indices and tables	15
	Python Module Index	17
	Index	19

1.1 Authentication

```
class accounts.authentication.EmailAuthBackend
```

```
    Bases: object
```

```
    Authenticate using e-mail address.
```

```
    authenticate (request, username=None, password=None)
```

```
    get_user (user_id)
```

```
class accounts.authentication.PhoneAuthBackend
```

```
    Bases: object
```

```
    Authenticate using phone number
```

```
    authenticate (request, 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 (request, 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': <django.forms.fields.CharField object>}

    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': <django.forms.fields.CharField object>}

    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': <phonenumbers_field.formfields.PhoneNumberField object>}

    clean_phone()

    declared_fields = {'phone': <phonenumbers_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', pre-
                                     fix=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', pre-
                                   fix=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 ForwardOne-  
        ToOneDescriptor 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.Verifier(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)
```

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 sub-
        classes.

    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`

a

- `accounts.activity_recorder_mixin`, 11
- `accounts.admin`, 12
- `accounts.authentication`, 1
- `accounts.decorators`, 11
- `accounts.device_generator`, 1
- `accounts.forms`, 2
- `accounts.models`, 4
- `accounts.signals`, 9
- `accounts.tokens`, 9
- `accounts.urls`, 9
- `accounts.utils`, 11
- `accounts.verifications`, 9
- `accounts.views`, 9

A

AccountActivationTokenGenerator (class in *accounts.tokens*), 9

accounts.activity_recorder_mixin (module), 11

accounts.admin (module), 12

accounts.authentication (module), 1

accounts.decorators (module), 11

accounts.device_generator (module), 1

accounts.forms (module), 2

accounts.models (module), 4

accounts.signals (module), 9

accounts.tokens (module), 9

accounts.urls (module), 9

accounts.utils (module), 11

accounts.verification (module), 9

accounts.views (module), 9

actions (*accounts.admin.CustomUserAdmin* attribute), 12

actions (*accounts.models.User* attribute), 7

activate() (in module *accounts.views*), 10

activate_account() (*accounts.admin.CustomUserAdmin* method), 12

activate_phone() (in module *accounts.views*), 10

Activity (class in *accounts.models*), 4

Activity.DoesNotExist, 4

Activity.MultipleObjectsReturned, 4

ActivityAdmin (class in *accounts.admin*), 12

alert_user() (in module *accounts.views*), 10

assign_permissions() (in module *accounts.utils*), 11

auth_guest() (in module *accounts.decorators*), 11

authenticate() (*accounts.authentication.EmailAuthBackend* method), 1

authenticate() (*accounts.authentication.PhoneAuthBackend* method), 1

authenticate() (*accounts.authentication.UsernameAuthBackend* method), 1

B

base_fields (*accounts.forms.LoginForm* attribute), 2

base_fields (*accounts.forms.PhoneVerificationForm* attribute), 2

base_fields (*accounts.forms.ProfileEditForm* attribute), 2

base_fields (*accounts.forms.TokenVerificationForm* attribute), 3

base_fields (*accounts.forms.TrustedDeviceForm* attribute), 3

base_fields (*accounts.forms.UserEditForm* attribute), 3

base_fields (*accounts.forms.UserRegistrationForm* attribute), 4

browser (*accounts.models.Device* attribute), 5

C

check_device_signature() (*accounts.models.Device* method), 5

check_token() (*accounts.verification.Verificator* method), 9

check_verified_fields() (*accounts.models.User* method), 7

clean() (*accounts.forms.LoginForm* method), 2

clean_password2() (*accounts.forms.UserRegistrationForm* method), 4

clean_phone() (*accounts.forms.PhoneVerificationForm* method), 2

create_action() (*accounts.activity_recorder_mixin.RecordsActivityMixin* static method), 11

create_action() (in module *accounts.utils*), 12

create_user_profile() (in module *accounts.signals*), 9

created (*accounts.models.Activity* attribute), 4
 created (*accounts.models.Device* attribute), 5
 CustomImageField (class in *accounts.utils*), 11
 CustomUserAdmin (class in *accounts.admin*), 12

D

date_hierarchy (*accounts.admin.LogEntryAdmin* attribute), 13
 date_of_birth (*accounts.models.Profile* attribute), 6
 declared_fields (*accounts.forms.LoginForm* attribute), 2
 declared_fields (*accounts.forms.PhoneVerificationForm* attribute), 2
 declared_fields (*accounts.forms.ProfileEditForm* attribute), 3
 declared_fields (*accounts.forms.TokenVerificationForm* attribute), 3
 declared_fields (*accounts.forms.TrustedDeviceForm* attribute), 3
 declared_fields (*accounts.forms.UserEditForm* attribute), 3
 declared_fields (*accounts.forms.UserRegistrationForm* attribute), 4
 Device (class in *accounts.models*), 5
 Device.DoesNotExist, 5
 Device.MultipleObjectsReturned, 5
 device_set (*accounts.models.User* attribute), 7
 DeviceAdmin (class in *accounts.admin*), 13
 DeviceInlineAdmin (class in *accounts.admin*), 13
 dispatch() (*accounts.activity_recorder_mixin.RecordsActivityMixin* method), 11

E

email_verified (*accounts.models.Profile* attribute), 6
 email_verified() (*accounts.admin.CustomUserAdmin* method), 12
 EmailAuthBackend (class in *accounts.authentication*), 1
 extra (*accounts.admin.DeviceInlineAdmin* attribute), 13

F

fetch_user_phone_from_session() (*accounts.verification.Verificator* method), 9
 fields (*accounts.forms.PhoneVerificationForm.Meta* attribute), 2
 fields (*accounts.forms.ProfileEditForm.Meta* attribute), 2

fields (*accounts.forms.TrustedDeviceForm.Meta* attribute), 3
 fields (*accounts.forms.UserEditForm.Meta* attribute), 3
 fields (*accounts.forms.UserRegistrationForm.Meta* attribute), 4
 fieldsets (*accounts.admin.CustomUserAdmin* attribute), 13
 form (*accounts.views.LoginView* attribute), 9
 form (*accounts.views.RegisterView* attribute), 10

G

generate_new_signature() (*accounts.models.Device* method), 5
 generate_token() (*accounts.verification.Verificator* method), 9
 get() (*accounts.views.LoginView* method), 10
 get() (*accounts.views.ProfileEditView* method), 10
 get() (*accounts.views.RegisterView* method), 10
 get_absolute_url() (*accounts.models.Device* method), 5
 get_absolute_url() (*accounts.models.User* method), 7
 get_ip() (in module *accounts.device_generator*), 1
 get_location() (in module *accounts.device_generator*), 1
 get_next_by_created() (*accounts.models.Activity* method), 4
 get_next_by_created() (*accounts.models.Device* method), 5
 get_next_by_date_joined() (*accounts.models.User* method), 7
 get_previous_by_created() (*accounts.models.Activity* method), 4
 get_previous_by_created() (*accounts.models.Device* method), 5
 get_previous_by_date_joined() (*accounts.models.User* method), 7
 get_user() (*accounts.authentication.EmailAuthBackend* method), 1
 get_user() (*accounts.authentication.PhoneAuthBackend* method), 1
 get_user() (*accounts.authentication.UsernameAuthBackend* method), 1
 get_user_agent() (in module *accounts.device_generator*), 1
 groups (*accounts.models.User* attribute), 7

H

has_add_permission() (*accounts.admin.LogEntryAdmin* method), 13
 has_change_permission() (*accounts.admin.LogEntryAdmin* method), 13

[has_delete_permission\(\)](#) ([accounts.admin.LogEntryAdmin](#) method), 13
[has_view_permission\(\)](#) ([accounts.admin.LogEntryAdmin](#) method), 14

I

[id](#) ([accounts.models.Activity](#) attribute), 4
[id](#) ([accounts.models.Device](#) attribute), 5
[id](#) ([accounts.models.Profile](#) attribute), 6
[id](#) ([accounts.models.User](#) attribute), 8
[inlines](#) ([accounts.admin.CustomUserAdmin](#) attribute), 13
[ip](#) ([accounts.models.Device](#) attribute), 5
[is_valid\(\)](#) ([accounts.forms.TokenVerificationForm](#) method), 3

L

[last_login_at\(\)](#) ([accounts.admin.CustomUserAdmin](#) static method), 13
[last_login_location\(\)](#) ([accounts.admin.CustomUserAdmin](#) static method), 13
[list_display](#) ([accounts.admin.ActivityAdmin](#) attribute), 12
[list_display](#) ([accounts.admin.CustomUserAdmin](#) attribute), 13
[list_display](#) ([accounts.admin.DeviceAdmin](#) attribute), 13
[list_display](#) ([accounts.admin.LogEntryAdmin](#) attribute), 14
[list_filter](#) ([accounts.admin.ActivityAdmin](#) attribute), 12
[list_filter](#) ([accounts.admin.CustomUserAdmin](#) attribute), 13
[list_filter](#) ([accounts.admin.DeviceAdmin](#) attribute), 13
[list_filter](#) ([accounts.admin.LogEntryAdmin](#) attribute), 14
[list_select_related](#) ([accounts.admin.CustomUserAdmin](#) attribute), 13
[location](#) ([accounts.models.Device](#) attribute), 5
[logentry_set](#) ([accounts.models.User](#) attribute), 8
[LogEntryAdmin](#) (class in [accounts.admin](#)), 13
[login\(\)](#) ([accounts.forms.LoginForm](#) method), 2
[LoginForm](#) (class in [accounts.forms](#)), 2
[LoginView](#) (class in [accounts.views](#)), 9

M

[machine](#) ([accounts.models.Device](#) attribute), 6
[media](#) ([accounts.admin.ActivityAdmin](#) attribute), 12
[media](#) ([accounts.admin.CustomUserAdmin](#) attribute), 13
[media](#) ([accounts.admin.DeviceAdmin](#) attribute), 13
[media](#) ([accounts.admin.DeviceInlineAdmin](#) attribute), 13
[media](#) ([accounts.admin.LogEntryAdmin](#) attribute), 14
[media](#) ([accounts.admin.ProfileAdmin](#) attribute), 14
[media](#) ([accounts.forms.LoginForm](#) attribute), 2
[media](#) ([accounts.forms.PhoneVerificationForm](#) attribute), 2
[media](#) ([accounts.forms.ProfileEditForm](#) attribute), 3
[media](#) ([accounts.forms.TokenVerificationForm](#) attribute), 3
[media](#) ([accounts.forms.TrustedDeviceForm](#) attribute), 3
[media](#) ([accounts.forms.UserEditForm](#) attribute), 3
[media](#) ([accounts.forms.UserRegistrationForm](#) attribute), 4
[model](#) ([accounts.admin.ActivityAdmin](#) attribute), 12
[model](#) ([accounts.admin.DeviceAdmin](#) attribute), 13
[model](#) ([accounts.admin.DeviceInlineAdmin](#) attribute), 13
[model](#) ([accounts.admin.ProfileAdmin](#) attribute), 14
[model](#) ([accounts.forms.PhoneVerificationForm.Meta](#) attribute), 2
[model](#) ([accounts.forms.ProfileEditForm.Meta](#) attribute), 2
[model](#) ([accounts.forms.TrustedDeviceForm.Meta](#) attribute), 3
[model](#) ([accounts.forms.UserEditForm.Meta](#) attribute), 3
[model](#) ([accounts.forms.UserRegistrationForm.Meta](#) attribute), 4

N

[notify_user\(\)](#) ([accounts.models.Device](#) static method), 6

O

[object_link\(\)](#) ([accounts.admin.LogEntryAdmin](#) method), 14
[objects](#) ([accounts.models.Activity](#) attribute), 4
[objects](#) ([accounts.models.Device](#) attribute), 6
[objects](#) ([accounts.models.Profile](#) attribute), 6
[operating_system](#) ([accounts.models.Device](#) attribute), 6

P

[path\(\)](#) (in module [accounts.urls](#)), 9
[phone](#) ([accounts.models.User](#) attribute), 8
[phone_verified](#) ([accounts.models.Profile](#) attribute), 6
[phone_verified\(\)](#) ([accounts.admin.CustomUserAdmin](#) method), 13
[PhoneAuthBackend](#) (class in [accounts.authentication](#)), 1

PhoneVerificationForm (class in *accounts.forms*), 2
 PhoneVerificationForm.Meta (class in *accounts.forms*), 2
 photo (*accounts.models.Profile* attribute), 6
 post () (*accounts.views.LoginView* method), 10
 post () (*accounts.views.ProfileEditView* method), 10
 post () (*accounts.views.RegisterView* method), 10
 process_verification_request () (*accounts.verification.Verificator* method), 9
 profile (*accounts.models.User* attribute), 8
 Profile (class in *accounts.models*), 6
 profile () (in module *accounts.views*), 10
 Profile.DoesNotExist, 6
 Profile.MultipleObjectsReturned, 6
 profile_form (*accounts.views.ProfileEditView* attribute), 10
 ProfileAdmin (class in *accounts.admin*), 14
 ProfileEditForm (class in *accounts.forms*), 2
 ProfileEditForm.Meta (class in *accounts.forms*), 2
 ProfileEditView (class in *accounts.views*), 10

R

re_path () (in module *accounts.urls*), 9
 readonly_fields (*accounts.admin.ProfileAdmin* attribute), 14
 RecordsActivityMixin (class in *accounts.activity_recorder_mixin*), 11
 RegisterView (class in *accounts.views*), 10
 request_verification_email () (in module *accounts.views*), 10

S

save () (*accounts.models.User* method), 8
 save_form_data () (*accounts.utils.CustomImageField* method), 11
 search_fields (*accounts.admin.ActivityAdmin* attribute), 12
 search_fields (*accounts.admin.LogEntryAdmin* attribute), 14
 send_verification_email () (in module *accounts.utils*), 12
 send_verification_sms () (*accounts.verification.Verificator* method), 9
 suspend_account () (*accounts.admin.CustomUserAdmin* method), 13

T

target (*accounts.models.Activity* attribute), 4
 target_ct (*accounts.models.Activity* attribute), 4
 target_ct_id (*accounts.models.Activity* attribute), 5

target_id (*accounts.models.Activity* attribute), 5
 temp_token (*accounts.models.Profile* attribute), 6
 template_name (*accounts.views.LoginView* attribute), 10
 template_name (*accounts.views.ProfileEditView* attribute), 10
 template_name (*accounts.views.RegisterView* attribute), 10
 TokenVerificationForm (class in *accounts.forms*), 3
 trusted (*accounts.models.Device* attribute), 6
 TrustedDeviceForm (class in *accounts.forms*), 3
 TrustedDeviceForm.Meta (class in *accounts.forms*), 3

U

user (*accounts.models.Activity* attribute), 5
 user (*accounts.models.Device* attribute), 6
 user (*accounts.models.Profile* attribute), 7
 User (class in *accounts.models*), 7
 User.DoesNotExist, 7
 User.MultipleObjectsReturned, 7
 user_detail () (in module *accounts.views*), 10
 user_form (*accounts.views.ProfileEditView* attribute), 10
 user_id (*accounts.models.Activity* attribute), 5
 user_id (*accounts.models.Device* attribute), 6
 user_id (*accounts.models.Profile* attribute), 7
 user_permissions (*accounts.models.User* attribute), 8
 UserEditForm (class in *accounts.forms*), 3
 UserEditForm.Meta (class in *accounts.forms*), 3
 UsernameAuthBackend (class in *accounts.authentication*), 1
 UserRegistrationForm (class in *accounts.forms*), 3
 UserRegistrationForm.Meta (class in *accounts.forms*), 4

V

verb (*accounts.models.Activity* attribute), 5
 verbose_name (*accounts.admin.DeviceInlineAdmin* attribute), 13
 verbose_name_plural (*accounts.admin.DeviceInlineAdmin* attribute), 13
 Verificator (class in *accounts.verification*), 9
 verify () (*accounts.verification.Verificator* method), 9
 verify_phone () (in module *accounts.views*), 10

W

widgets (*accounts.forms.ProfileEditForm.Meta* attribute), 2

widgets (*accounts.forms.TrustedDeviceForm.Meta attribute*), [3](#)