Userena Documentation

Release 2.0.1

Petar Radosevic

Oct 02, 2017

Contents

This documentation covers 2.0.1 release of django-userena application. A Django application that takes care of your account needs.

CHAPTER 1

Why userena?

Because we have done the hard work for you. Userena supplies you with signup, signin, account editing, privacy settings and private messaging. All you have to do is plug it into your project and you will have created account management with the following options:

- User has to **activate** their account by clicking on a activation link in an email sent to them.
- **Permissions** for viewing, changing and deleting accounts is implemented on an user and object basis with the help of django-guardian.
- Optionally **secure** userena by using https. If you change the settings to use https, userena will switch to the secure protocol on it's views and emails.
- All **templates** are already supplied for you. Only override those that don't fit with your needs.
- Mugshots are supplied by **Gravatar** or uploaded by the user. The default mugshot can be set in the settings.
- **Messaging** system between users that either get's displayed as conversations (iPhone like) or sorted per subject (Gmail).

CHAPTER 2

Help out

Found a bug in userena? File an issue at Github. Have an improvement? Fork it and add it, or if you can't code it, contact us to do it.

CHAPTER 3

Deprecation warnigns

2.0.0 version:

- `userena.utils.get_user_model()` is deprecated and will be removed in version 3.0.0. Use `django.contrib.auth.get_user_model()`

CHAPTER 4

Changes and releases

For changes history and available releases see following pages on GitHub repository:

- UDATES.md
- releases

CHAPTER 5

Contents

Installation.

Before install django-userena, you'll need to have a copy of Django 1.5 or newer installed. django-userena is tested under Python 2.6, 2.7, 3.2, 3.3, 3.4, and 3.5 (all versions on which Django 1.5 and higher is declared to work)

For further information, consult the Django download page, which offers convenient packaged downloads and installation instructions.

Support for Django versions below 1.7

Starting from version 2.0.0 django-userena supports Django 1.9 release and drops the support for Django 1.4. It is tested and works for all releases from 1.5 to 1.9 but some older versions of Django require some additional work in order to ensure full compatibility:

- Django versions below 1.7 require South for data migrations. django-userena provides new-style migrations for built-in Django schema migrations engine (available starting from Django 1.7) but provides old South migrations in `userena.south_migrations` and `userena.contrib.umessages.south_migrations` subpackages. South (starting from version 1.0.0) should be able to pick them easily if you still use it even for Django versions 1.7 or greater. Anyway, South support in django-userena is deprecated and will be removed in some future major release (3.0.0 or 4.0.0 version).
- django-guardian is one of the main dependecies of django-userena and every release of this package seems to drop some bacwards compatibility without resonable versioning scheme. This is why for Django 1.5 and 1.6 you need to fix django-guardian on version 1.3.2 or lower manually.

Installing django-userena.

You can install django-userena automagically with `pip`. Or by manually placing it on on your `PYTHON_PATH`. The recommended way is the shown in *Automatic installation with pip.*.

It is also recommended to use virtualenv *to have an isolated python environment. This way it's possible to create a tailored environment for each project.*

Automatic installation with pip.

Automatic install with pip. All you have to do is run the following command:

```
pip install django-userena
```

If you want to have a specific version of userena, you can do so by adding the following:

```
pip install django-userena==1.0.1
```

Manual installation with easy_install.

Clone the Git repository from Github. Then you can direct easy_install to the `setup.py` file. For ex.:

```
git clone git://github.com/bread-and-pepper/django-userena.git
cd django-userena
easy_install setup.py
```

Automatic installation of development version with pip.

You can tell pip to install django-userena by supplying it with the git repository on Github. Do this by typing the following in your terminal:

```
pip install -e git+git://github.com/bread-and-pepper/django-userena.git#egg=userena
```

Manual installation of development version with git.

Clone userena with:

```
git clone git://github.com/bread-and-pepper/django-userena.git
```

You now have a directory `django-userena` which contains the `userena` application. You can add userena to your `$PYTHONPATH` by symlinking it. For example:

```
cd YOUR_PYTHON_PATH
ln -s ~/src/django-userena/userena userena
```

Now userena is available to your project.

Required settings

You need to make some changes Django settings if you want to use Userena in your project. This means modifying `AUTHENTICATION_BACKENDS`, `INSTALLED_APPS` and optionally `MIDDLEWARE_CLASSES`.

Begin by adding `userena`, `guardian` and `easy_thumbnails` to the `INSTALLED_APPS` in your settings.py file of your project. `django.contrib.sites` must also be present if it is not already (see Django docs.).

Next add `UserenaAuthenticationBackend` and `ObjectPermissionBackend` also in your settings.py file, from django-guardian, at the top of `AUTHENTICATION_BACKENDS`. If you only have Django's default backend, adding django-guardian and that of userena will get the following:

```
AUTHENTICATION_BACKENDS = (
    'userena.backends.UserenaAuthenticationBackend',
    'guardian.backends.ObjectPermissionBackend',
    'django.contrib.auth.backends.ModelBackend',
)
```

Start New App

Next, you need to create a new app on your Django project. In your Command Prompt shell, type: `python manage.py startapp accounts`. We are creating a new app for Userena titled 'accounts'.

Next, add `accounts` to the `INSTALLED_APPS` in your settings.py file.

Email Backend

Userena uses the Django email facilities to send mail to users, for example after user signup for email verification. By default Django uses the SMTP backend, which may cause issues in development and/or if the default SMTP settings are not suitable for your environment. It is recommended to explicitly set the email backend provider in your settings.py. For example:

```
EMAIL_BACKEND = 'django.core.mail.backends.dummy.EmailBackend'
```

To use GMail SMTP, you may use the following code in your settings.py:

```
EMAIL_USE_TLS = True
EMAIL_HOST = 'smtp.gmail.com'
EMAIL_PORT = 587
EMAIL_HOST_USER = 'yourgmailaccount@gmail.com'
EMAIL_HOST_PASSWORD = 'yourgmailpassword'
```

See: Django Email Documentation

Profiles

Userena needs you to define the profile that is used by supplying Django's `AUTH_PROFILE_MODULE` setting. Userena supplies the following two base profiles for you that you should use for your own profile model by inheriting from them:

`UserenaBaseProfile` Basic profile that supplies your user with mugshots and the necessary fields for privacy settings.

`UserenaLanguageBaseProfile` Adds an extra field that lets the user define its preferred language after logging in to your site.

IMPORTANT: The above profiles are `abstract` models. This means that you cannot use them directly in `AUTH_PROFILE_MODULE` but you must create your own profile model which inherits from one of the above models. This models must also connect itself to the `User` model of Django.

```
from django.contrib.auth.models import User
from django.utils.translation import ugettext as _
from userena.models import UserenaBaseProfile

class MyProfile(UserenaBaseProfile):
    user = models.OneToOneField(User,
                                unique=True,
```

```
                                verbose_name=_('user'),
                                related_name='my_profile')
    favourite_snack = models.CharField(_('favourite snack'),
                                       max_length=5)
```

If you want the user have the ability to choose their default language in their profile, you must add `userena.middleware.UserenaLocaleMiddleware` at the end of `MIDDLEWARE_CLASSES` in your Django settings. This does require a profile model which has a language field. You can use the `UserenaLanguageBaseProfile` class of userena that does this for you.

The URI's

Userena has a `URLconf` which sets all the urls and views for you. This should be included in your project's root `URLconf`.

For example, to place the URIs under the prefix `/accounts/`, you could add the following to your project's root `URLconf`. Add this code under `urlpatterns` in your urls.py file.

```
(r'^accounts/', include('userena.urls')),
```

This should have you a working accounts application for your project. See the *settings* for further configuration options.

Required settings

Django-guardian requires you to set the `ANONYMOUS_USER_ID` setting. I always set this to `-1`. As noted before, you are also required to set the `AUTH_PROFILE_MODULE` to your custom defined profile.

For example, add the following into your settings.py file:

```
ANONYMOUS_USER_ID = -1

AUTH_PROFILE_MODULE = 'accounts.MyProfile'
```

To integrate Django with userena you should alter the following three settings to reflect the URI you have chosen for userena. For example, if userena lives under `accounts`:

```
USERENA_SIGNIN_REDIRECT_URL = '/accounts/%(username)s/'
LOGIN_URL = '/accounts/signin/'
LOGOUT_URL = '/accounts/signout/'
```

The above should supply you with a fully functional account management app for your project. You can look into the next chapter to fully customize userena to your likings.

To integrate Userena with your domain you must create a Site for it in the Django admin screen (e.g. http://<yoursite.com>/admin/sites/) and then put the id for that site in the SITE_ID setting variable.:

To look up your site_id open a shell in manage.py (manage.py shell) and:

Set SITE_ID to the id of the desired name.

Permission check

Sometimes Django decides to skip installing the default permissions for a model. To check if all permissions are there, run the `check_permissions` in the management *Commands.*.

Settings

Userena comes with a few settings that enables you to tweak the user experience for you users. There are also a few Django settings that are relevant for Userena.

Userena settings

USERENA_SIGNIN_AFTER_SIGNUP

Default `False` (integer)

Boolean that defines if a user should be logged in after a successful sign up.

If True, USERENA_ACTIVATION_REQUIRED must be False for the sign-in to happen.

Note that USERENA_SIGNIN_REDIRECT_URL will not be respected for the automatic sign-in. The user will be redirect to the value of 'success_url' in userena.views.signup.

You can override 'success_url' in your urls.py. See the "How do I add extra fields to forms?" example in the FAQ, where the 'signup_form' variable is overridden.

USERENA_SIGNIN_REDIRECT_URL

Default `/accounts/%(username)s/'` (string)

A string which defines the URI where the user will be redirected to after signin.

USERENA_ACTIVATION_REQUIRED

Default: `True` (integer)

Boolean that defines if a activation is required when creating a new user.

USERENA_ACTIVATION_DAYS

Default: `7` (integer)

A integer which stands for the amount of days a user has to activate their account. The user will be deleted when they still haven't activated their account after these amount of days by running the `cleanexpired` *command*.

USERENA_ACTIVATION_NOTIFY

Default: `True` (boolean)

A boolean that turns on/off the sending of a notification when `USERENA_ACTIVATION_NOTIFY_DAYS` away the activation of the user will expire and the user will be deleted.

USERENA_ACTIVATION_NOTIFY_DAYS

Default: `2` (integer)

The amount of days, before the expiration of an account, that a notification get's send out. Warning the user of his coming demise.

USERENA_ACTIVATED

Default: `ALREADY_ACTIVATED` (string)

String that defines the value that the `activation_key` will be set to after a successful signup.

USERENA_REMEMBER_ME_DAYS

Default: `(gettext('a month'), 30))` (tuple)

A tuple containing a string and an integer which stand for the amount of days a user can choose to be remembered by your project. The string is the human readable version that gets displayed in the form. The integer stands for the amount of days that this string represents.

USERENA_FORBIDDEN_USERNAMES

Default: `('signup', 'signout', 'signin', 'activate', 'me', 'password')` (tuple)

A tuple containing the names which cannot be used as username in the signup form.

USERENA_MUGSHOT_GRAVATAR

Default: `True` (boolean)

A boolean defining if mugshots should fallback to Gravatar service when no mugshot is uploaded by the user.

USERENA_MUGSHOT_GRAVATAR_SECURE

Default: `USERENA_USE_HTTPS` (boolean)

A boolean defining if the secure URI of Gravatar is used. Defaults to the same value as `USERENA_USE_HTTPS`.

USERENA_MUGSHOT_DEFAULT

Default: `identicon` (string)

A string for the default image used when no mugshot is found. This can be either a URI to an image or if *USERENA_MUGSHOT_GRAVATAR* is `True` one of the following options:

`404` Do not load any image if none is associated with the email hash, instead return an HTTP 404 (File Not Found) response.

`mm` Mystery-man, a simple, cartoon-style silhouetted outline of a person (does not vary by email hash).

`identicon` A geometric pattern based on an email hash.

`monsterid` A generated 'monster' with different colors, faces, etc.

`wavatar` Generated faces with differing features and backgrounds

USERENA_MUGSHOT_SIZE

Default: `80` (int)

Integer defining the size (in pixels) of the sides of the mugshot image.

USERENA_MUGSHOT_PATH

Default: `mugshots/` (string)

The default path that the mugshots will be saved to. Is appended to the `MEDIA_PATH` in your Django settings.

You can use the following options as arguments (f.ex. `mugshots/%(username)s/`):

`id` User.id

`username` User.username

`date` User.date_joined

`date_now` Current date

USERENA_USE_HTTPS

Default: `False` (boolean)

Boolean that defines if you have a secure version of your website. If so, userena will redirect sensitive URI's to the secure protocol.

USERENA_DEFAULT_PRIVACY

Default: `registered` (string)

Defines the default privacy value for a newly registered user. There are three options:

`closed` Only the owner of the profile can view their profile.

`registered` All registered users can view their profile.

`open` All users (registered and anonymous) can view their profile.

USERENA_PROFILE_DETAIL_TEMPLATE

Default: `userena/profile_detail.html` (string)

Template to use for rendering user profiles. This allows you to specify a template in your own project which extends `userena/profile_detail.html`.

USERENA_PROFILE_LIST_TEMPLATE

Default: `userena/profile_list.html` (string)

Template to use for rendering users list. This allows you to specify a template in your own project which extends `userena/profile_list.html`.

USERENA_DISABLE_PROFILE_LIST

Default: `False` (boolean)

Boolean value that defines if the `profile_list` view is enabled within the project. If so, users can view a list of different profiles.

USERENA_DISABLE_SIGNUP

Default: `False` (boolean)

Boolean value that defines if signups are disabled within the project. If so, users trying to sign up will be denied.

USERENA_USE_MESSAGES

Default: `True` (boolean)

Boolean value that defines if userena should use the django messages framework to notify the user of any changes.

USERENA_LANGUAGE_FIELD

Default: `language` (string)

The language field that is used in the custom profile to define the preferred language of the user.

USERENA_WITHOUT_USERNAMES

Default: `False` (boolean)

Defines if usernames are used within userena. Currently it's often for the users convenience that only an email is used for identification. With this setting you get just that.

USERENA_HIDE_EMAIL

Default: `False` (boolean)

Prevents email addresses from being displayed to other users if set to `True`.

USERENA_HTML_EMAIL

Default: `False` (boolean)

If `True` multipart emails are generated using html templates.

USERENA_USE_PLAIN_TEMPLATE

Default: `True` (boolean)

Uses a text template for plain text part (when `USERENA_HTML_EMAIL = True`). When `USERENA_HTML_EMAIL = False`, plain text templates are always used for emails even if `USERENA_USE_PLAIN_TEMPLATE = False`.

USERENA_REGISTER_PROFILE

Default: `True` (boolean)

If `True` userena will register the profile model with Django Admin for you. It uses a GuardedModelAdmin when registering. This allows per user object permissions to be set via the admin. If `False` you will have to regiter the profile with the Django Admin yourself.

USERENA_REGISTER_USER

Default: `True` (boolean)

If `True` userena will first unregister the user model with the admin and then reregister the user model using a GuardedModelAdmin. This allows you to set per user object permissons. If `False` and you want to you set per user object permissions on the user model via the admin you will have to unregister and reregiter the user model with the Django Admin yourself.

Django settings

LOGIN_URL

Default: `/accounts/login/` (string)

The URL where requests are redirected for login, especially when using the login_required() decorator.

In userena this URI normally would be `/accounts/signin/`.

LOGOUT_URL

Default: `/accounts/logout/` (string) LOGIN_URL counterpart.

In userena this URI normally would be `/accounts/signout/`.

LOGIN_REDIRECT_URL

Default: `/accounts/profile/`

In userena this URI should point to the profile of the user. Thus a string of `/accounts/%(username)s/` is best.

AUTH_PROFILE_MODULE

Default: `not defined`

This should point to the model that is your custom made profile.

Signals

Userena contains a few signals which you can use in your own application if you want to have custom actions when a account get's changed. All signals are located in `userena/signals.py` file.

signup_complete

This signal get's fired when an user signs up at your site. Note: This doesn't mean that the user is activated. The signal provides you with the `user` argument which Django's `User` class.

activation_complete

A user has succesfully activated their account. The signal provides you with the `user` argument which Django's `User` class.

confirmation_complete

A user has succesfully changed their email. The signal provides you with the `user` argument which Django's `User` class, and the `old_email` argument which is the user's old email address as a string.

password_complete

A user has succesfully changed their password. The signal provides you with the `user` argument which Django's `User` class.

Commands.

Userena currently comes with two commands. `cleanexpired` for cleaning out the expired users and `check_permissions` for checking the correct permissions needed by userena.

Clean expired

Search for users that still haven't verified their e-mail address after `USERENA_ACTIVATION_DAYS` and delete them. Run by

```
./manage.py clean_expired
```

Check permissions

This command shouldn't be run as a cronjob. This is only for emergency situations when some permissions are not correctly set for users. For example when userena get's implemented in an already existing project. Run by

```
./manage.py check_permissions
```

F.A.Q

I get a "`Permission matching query does not exist`" exception

Sometimes Django decides not to install the default permissions for a model and thus the `change_profile` permission goes missing. To fix this, run the `check_permissions` in *Commands.*. This checks all permissions and adds those that are missing.

I get a "`Site matching query does not exist.`" exception

This means that your settings.SITE_ID value is incorrect. See the instructions on SITE_ID in the [Installation section](http://docs.django-userena.org/en/latest/installation.html).

<ProfileModel> is already registered exception

Userena already registered your profile model for you. If you want to customize the profile model, you can do so by registering your profile as follows:

```
# Unregister userena's
admin.site.unregister(YOUR_PROFILE_MODEL)

# Register your own admin class and attach it to the model
admin.site.register(YOUR_PROFILE_MODEL, YOUR_PROFILE_ADMIN)
```

Can I still add users manually?

Yes, but Userena requires there to be a *UserenaSignup* object for every registered user. If it's not there, you could receive the following error:

```
Exception Type: DoesNotExist at /accounts/mynewuser/email/
```

So, whenever you are manually creating a user (outside of Userena), don't forget to also create a *UserenaSignup* object.

How can I have multiple profiles per user?

One way to do this is by overriding the *save* method on *SignupForm* with your own form, extending userena's form and supply this form with to the signup view. For example:

```
def save(self):
    """ My extra profile """
    # Let userena do it's thing
    user = super(SignupForm, self).save()

    # You do all the logic needed for your own extra profile
    custom_profile = ExtraProfile()
    custom_profile.extra_field = self.cleaned_data['field']
    custom_profile.save()

    # Always return the new user
    return user
```

Important to note here is that you should always return the newly created *User* object. This is something that userena expects. Userena will take care of creating the user and the "standard" profile.

Don't forget to supply your own form to the signup view by overriding the URL in your *urls.py*:

```
(r'^accounts/signup/$',
 'userena.views.signup',
 {'signup_form': SignupExtraProfileForm}),
```

How do I add extra fields to forms?

This is done by overriding the default templates. A demo tells more than a thousand words. So here's how you add the first and last name to the signup form. First you override the signup form and add the fields.

```
from django import forms
from django.utils.translation import ugettext_lazy as _

from userena.forms import SignupForm

class SignupFormExtra(SignupForm):
    """
    A form to demonstrate how to add extra fields to the signup form, in this
    case adding the first and last name.


    """
    first_name = forms.CharField(label=_(u'First name'),
                                 max_length=30,
                                 required=False)

    last_name = forms.CharField(label=_(u'Last name'),
                                max_length=30,
                                required=False)

    def __init__(self, *args, **kw):
        """

        A bit of hackery to get the first name and last name at the top of the
        form instead at the end.

        """
        super(SignupFormExtra, self).__init__(*args, **kw)
        # Put the first and last name at the top
        new_order = self.fields.keyOrder[:-2]
        new_order.insert(0, 'first_name')
        new_order.insert(1, 'last_name')
        self.fields.keyOrder = new_order

    def save(self):
        """
        Override the save method to save the first and last name to the user
        field.

        """
        # First save the parent form and get the user.
        new_user = super(SignupFormExtra, self).save()

        # Get the profile, the `save` method above creates a profile for each
        # user because it calls the manager method `create_user`.
        # See: https://github.com/bread-and-pepper/django-userena/blob/master/userena/
↪managers.py#L65
        user_profile = new_user.get_profile()

        user_profile.first_name = self.cleaned_data['first_name']
        user_profile.last_name = self.cleaned_data['last_name']
        user_profile.save()

        # Userena expects to get the new user from this form, so return the new
        # user.
        return new_user
```

Finally, to use this form instead of our own, override the default URI by placing a new URI above it.

```
(r'^accounts/signup/$',
 'userena.views.signup',
 {'signup_form': SignupFormExtra}),
```

That's all there is to it!

API Reference

Backends

Return to *API Reference*.

Decorators

Return to *API Reference*.

secure_required

userena.decorators.**secure_required**(*view_func*)

Decorator to switch an url from http to https.

If a view is accessed through http and this decorator is applied to that view, than it will return a permanent redirect to the secure (https) version of the same view.

The decorator also must check that USERENA_USE_HTTPS is enabled. If disabled, it should not redirect to https because the project doesn't support it.

Forms

Return to *API Reference*.

SignupForm

SignupFormOnlyEmail

SignupFormTos

AuthenticationForm

ChangeEmailForm

EditProfileForm

Managers

Return to *API Reference*

UserenaManager

UserenaBaseProfileManager

Middleware

Return to *API Reference*

UserenaLocaleMiddleware

Models

Return to *API Reference*.

upload_to_mugshot

UserenaSignup

UserenaBaseProfile

UserenaLanguageBaseProfile

Utils

get_gravatar

signin_redirect

generate_sha1

get_profile_model

Views

signup

activate

email_confirm

direct_to_user_template

signin

email_change

password_change

profile_edit

profile_detail

profile_list

Contrib: uMessages

uMessages

Userena's umesagges supplies you with iPhone like messaging system for your users.

Installation

You install it by adding `userena.contrib.umessages` to your `INSTALLED_APPS` setting. You also need to add it to your urlconf. For example:

```
(r'^messages/', include('userena.contrib.umessages.urls')),
```

A `syncdb` later and you have a great messaging system for in your application.

API Reference

Managers

MessageManager

class `userena.contrib.umessages.managers.`**MessageManager**
Manager for the `Message` model.

get_conversation_between(*um_from_user*, *um_to_user*)
Returns a conversation between two users

send_message(*sender*, *um_to_user_list*, *body*)
Send a message from a user, to a user.

Parameters

- **sender** – The `User` which sends the message.
- **um_to_user_list** – A list which elements are `User` to whom the message is for.
- **message** – String containing the message.

Views

MessageListView

MessageDetailListView

message_compose

message_remove

CHAPTER 6

Indices and tables

- genindex
- search

Python Module Index

u

Index

G

M

S

U