django-rest-email-auth Documentation

Release 2.1.0

Chathan Driehuys

Dec 13, 2019

Contents

CHAPTER 1

Important Links

Project Homepage https://github.com/cdriehuys/django-rest-email-auth

Documentation https://django-rest-email-auth.readthedocs.io

CHAPTER 2

Overview

This package provides a simple way to enable authentication with email addresses for a REST API.

Features:

- Verification of user email addresses
- Authentication using any of a user's verified email addresses
- Registration of new users
- Password resets using verified emails

CHAPTER 3

Adding Features

This package was primarily developed for a single project's use. As such it may lack features that other projects require. If that is the case, please open an issue and let us know.

CHAPTER 4

License

This project is licensed under the MIT License.

CHAPTER 5

Authors

Chathan Driehuys (chathan@driehuys.com)

5.1 Installation

5.1.1 Requirements

- Python 3.6 or 3.7
- Django 1.11, 2.0, or 2.1. Other versions may work, but they are not officially supported.

5.1.2 Getting the Package

The easiest way to install the package is with pip.

To get the most recent release:

```
$ pip install django-rest-email-auth
```

To get the most recent development build:

```
$ pip install git+https://github.com/cdriehuys/django-rest-email-auth.git#egg=django-
↪rest-email-auth
```

5.1.3 Required Configuration

In `settings.py`, make sure the following settings are present:

```
INSTALLED_APPS = [
    # At least these default Django apps must be installed:
    'django.contrib.auth',
    'django.contrib.contenttypes',

    # DRF must be listed for the browseable API to work
    'rest_framework',

    # Finally, the app itself
    'rest_email_auth',
]


AUTHENTICATION_BACKENDS = [
    'rest_email_auth.authentication.VerifiedEmailBackend',
    'django.contrib.auth.backends.ModelBackend',
]


# The minimal settings dict required for the app
REST_EMAIL_AUTH = {
    'EMAIL_VERIFICATION_URL': 'https://example.com/verify/{key}',
    'PASSWORD_RESET_URL': 'https://example.com/reset/{key}',
}
```

Email Setup

In addition to the above settings, we also require that Django be configured to send emails. Please configure any of the `EMAIL_*` settings that apply to your setup. See Django's email settings for more information.

URLs

After the settings have been configured, include the app's URLs in `urls.py`:

```
from django.urls import include, path
# If you're using Django 1.11:
# from django.conf.urls import include, url


urlpatterns = [
    path('account/', include('rest_email_auth.urls')),

    # If you're using Django 1.11:
    # url(r'account/', include('rest_email_auth.urls')),
]
```

5.1.4 Post-Installation

After the app has been installed and configured, its migrations must be run:

```
$ manage.py migrate
```

5.2 Configuration

Some behaviors can be controlled by setting options in `settings.py`. To override a setting, add it to the `REST_EMAIL_AUTH` dictionary in the settings file. For example:

```
# settings.py

REST_EMAIL_AUTH = {
    'EMAIL_VERIFICATION_URL': 'https://example.com/verify/{key}'
}
```

5.2.1 Required Settings

These settings must be configured before the app will function.

EMAIL_VERIFICATION_URL

A template for constructing the URL where a user can verify their email address. This URL is sent to the user in the verification email after they add a new email address. Any instance of `{key}` in the string will be replaced with the key required to confirm that user's email.

PASSWORD_RESET_URL

A template for constructing the URL where a user can reset their password. This URL is sent to the user at the email address specified when the password reset was requested. Any instance of `{key}` in the string will be replaced with the key required to complete the password reset.

5.2.2 Additional Settings

CONFIRMATION_EXPIRATION

Default: `datetime.timedelta(days=1)`

The duration that an email confirmation key is valid for.

CONFIRMATION_SAVE_PERIOD

Default: `datetime.timedelta(days=7)`

The duration of the grace period after a confirmation expires before it will be deleted by the `cleanemailconfirmations` command. See *Cleaning Expired Email Confirmations* for more information.

EMAIL_VERIFICATION_PASSWORD_REQUIRED

Default: `True`

A boolean indicating if users must provide their password when verifying an email address. Enabling this setting provides slightly more security at the cost of a decreased user experience.

Note: This setting was added in v2.0.0. To maintain backwards compatibility, it is enabled by default.

REGISTRATION_SERIALIZER

Default: `'rest_email_auth.serializers.RegistrationSerializer'`

The path to the serializer class that will be used for the registration endpoint. See *Registration Serializer* for more information.

PATH_TO_RESET_EMAIL_TEMPLATE

Default: `rest_email_auth/emails/reset-password`

The path to the template for the reset password email.

PATH_TO_VERIFY_EMAIL_TEMPLATE

Default: `rest_email_auth/emails/verify-email`

The path to the template for the verify user email.

PATH_TO_DUPLICATE_EMAIL_TEMPLATE

Default: `rest_email_auth/emails/duplicate-email`

The path to the template for the email that notifies of a duplicate email.

5.3 Endpoints

All endpoints here are relative to the path the app's URLs are included under.

5.3.1 Registering a New User

Once a user has submitted the registration form successfully, they will receive an email to verify their email address.

Due to privacy concerns related to exposing email addresses, a registration request with an email that is already in use will not fail. Instead it will send an email to the previously registered address notifying them of the registration attempt.

POST /register/

Request JSON Object

- **<User.USERNAME_FIELD>** (*string*) – The user's username.
- **email** (*string*) – The user's email address.
- **password** (*string*) – The user's password.

Response JSON Object

- **<User.USERNAME_FIELD>** (*string*) – The username the user was created with.

- **email** (*string*) – The email address the verification email was sent to.

Status Codes

- 201 Created – The request was successful and an email has been sent to the provded address.
- 400 Bad Request – An invalid request was made. Check the response data for details.

5.3.2 Resending a Verification Email

If a user lost the original verification email or it has expired, this endpoint can be used to send a new verification.

In order to avoid exposing email addresses, submitting an email to this endpoint that is not in the database will appear to be a successful request but is actually a no-op.

POST /resend-verification/

Request JSON Object

- **email** (*string*) – The address to resend a verification email to.

Response JSON Object

- **email** (*string*) – The address the new verification email was sent to.

Status Codes

- 200 OK – The request was succesful.
- 400 Bad Request – An invalid request was made. Check the response data for details.

5.3.3 Verify an Email Address

After a user receives a verification key, this endpoint is used to verify the email address.

POST /verify-email/

Request JSON Object

- **key** (*string*) – The verification key the user received.
- **password** (*string*) – The user's password. This field is only required if *EMAIL_VERIFICATION_PASSWORD_REQUIRED* is set to `True`.

Response JSON Object

- **email** (*string*) – The email address that was verified.

Status Codes

- 200 OK – The returned email was successfully verified.
- 400 Bad Request – An invalid request was made. Check the response data for details.

5.3.4 Listing or Creating Email Addresses

All the email addresses associated with a user can be listed using the following endpoint. This endpoint can also be used to add a new email address to the user's account.

GET /emails/

List the email addresses associated with the requesting user.

Response JSON Array of Objects

- **id** (*int*) – The ID that uniquely identifies the email address.
- **created_at** (*string*) – A timestamp identifying when the email address was added by the user.
- **email** (*string*) – The email's actual address.
- **is_primary** (*boolean*) – A boolean indicating if the address is the user's primary address.
- **is_verified** (*boolean*) – A boolean indicating if the email address has been verified.

POST /emails/

Add a new email address for the requesting user.

Request JSON Object

- **email** (*string*) – The address of the email to add.

Response JSON Object

- **id** (*int*) – The ID that uniquely identifies the email address.
- **created_at** (*string*) – A timestamp identifying when the email address was added by the user.
- **email** (*string*) – The email's actual address.
- **is_primary** (*boolean*) – A boolean indicating if the address is the user's primary address.
- **is_verified** (*boolean*) – A boolean indicating if the email address has been verified.

5.3.5 Viewing, Modifying, or Deleting a Specific Email Address

GET /emails/(int: *id*)/

Retrieve information about a specific email address.

Parameters

- **id** (*int*) – The unique ID of the email address to retrieve.

Response JSON Object

- **id** (*int*) – The ID that uniquely identifies the email address.
- **created_at** (*string*) – A timestamp identifying when the email address was added by the user.
- **email** (*string*) – The email's actual address.
- **is_primary** (*boolean*) – A boolean indicating if the address is the user's primary address.
- **is_verified** (*boolean*) – A boolean indicating if the email address has been verified.

Status Codes

- 200 OK – The email address was successfully retrieved.
- 404 Not Found – There is no email address with the provided *id* accessible to the requesting user.

PUT /emails/(int: *id*)/

Update a specific email address.

Parameters

- **id** (*int*) – The unique ID of the email address to retrieve.

Request JSON Object

- **email** (*string*) – The original email address. This field may not be changed.
- **is_primary** (*boolean*) – A boolean indicating if this address should be the user's primary email. This may only be `true` for a verified email.

Response JSON Object

- **id** (*int*) – The ID that uniquely identifies the email address.
- **created_at** (*string*) – A timestamp identifying when the email address was added by the user.
- **email** (*string*) – The email's actual address.
- **is_primary** (*boolean*) – A boolean indicating if the address is the user's primary address.
- **is_verified** (*boolean*) – A boolean indicating if the email address has been verified.

Status Codes

- 200 OK – The email address was successfully updated.
- 404 Not Found – There is no email address with the provided *id* accessible to the requesting user.

PATCH /emails/(int: *id*)/

Partially update a specific email address.

Parameters

- **id** (*int*) – The unique ID of the email address to retrieve.

Request JSON Object

- **email** (*string*) – *(Optional)* The original email address. This field may not be changed.
- **is_primary** (*boolean*) – *(Optional)* A boolean indicating if this address should be the user's primary email. This may only be `true` for a verified email.

Response JSON Object

- **id** (*int*) – The ID that uniquely identifies the email address.
- **created_at** (*string*) – A timestamp identifying when the email address was added by the user.
- **email** (*string*) – The email's actual address.
- **is_primary** (*boolean*) – A boolean indicating if the address is the user's primary address.
- **is_verified** (*boolean*) – A boolean indicating if the email address has been verified.

Status Codes

- 200 OK – The email address was successfully updated.
- 404 Not Found – There is no email address with the provided *id* accessible to the requesting user.

DELETE /emails/(**int:** *id*)**/**
Delete the email address with the specified *id*.

Parameters

- **id** (*int*) – The unique ID of the email address to delete.

Status Codes

- 204 No Content – The email address was successfully deleted.
- 404 Not Found – There is no email address with the provided *id* accessible to the requesting user.

5.3.6 Password Resets

Users may request a password reset using any of their verified emails.

Request a Reset

Sending a request to this endpoint will email the user a link that they can use to reset their password.

POST /request-password-reset/
Request a new password reset.

Request JSON Object

- **email** (*string*) – The email address to send the reset token to.

Status Codes

- 200 OK – This status is always returned to avoid leaking information about which emails exist in the system.

Reseting a Password

After the user receives an email address with a token they can use to reset their password, this endpoint should be used.

POST /reset-password/
Reset the user's password.

Request JSON Object

- **key** (*string*) – The token that the user was emailed authorizing the reset.
- **password** (*string*) – The user's new password.

Status Codes

- 200 OK – The user's password was reset successfully.
- 400 Bad Request – Either the provided key does not exist or has expired, or the provided password is invalid.

5.4 Commands

The app contains some additional commands that can be run from `manage.py`.

5.4.1 Cleaning Expired Email Confirmations

Command: `cleanemailconfirmations`

This command deletes any old, expired email confirmations. The duration that an expired confirmation will be saved is specified in the *CONFIRMATION_SAVE_PERIOD* setting.

5.5 Custom Serializers

5.5.1 Registration Serializer

It is fairly common to create a custom user model for a Django project. To accomodate this, you can easily substitute in a custom registration serializer that includes your added fields using the *REGISTRATION_SERIALIZER* setting.

To specify additional fields, simply subclass the provided registration serializer as follows:

```
from rest_email_auth.serializers import RegistrationSerializer


class MyRegistrationSerializer(RegistrationSerializer):

    class Meta:
        # You must include the 'email' field for the serializer to work.
        fields = (
            MyUserModel.USERNAME_FIELD,
            'password',
            'email',
            'other',
            'required',
            'fields',
        )
        model = MyUserModel
```

The provided registration serializer handles the conditional creation of the user and sending out a confirmation email. If you need more flexibility than is achievable by overriding the `Meta` class, you will have to dig in to the source code of the provided serializer. Feel free to create an issue if you have difficulties with this process.

5.6 Advanced Usage

5.6.1 Signals

Signals allow other applications to hook into rest-email-auth and add custom behaviors for events from the app.

The signals referenced below are importable from `rest_email_auth.signals`.

Email Verification

Signal Name: `email_verified`

Signal Arguments:

- **email:** The `EmailAddress` instance that was verfied.

This signal is triggered after an email address is verifed using an `EmailConfirmation` instance. Unless you have added custom logic allowing email addresses to be unverified, you can assume that the instance will be marked as verified after this point.

Registration

Signal Name: `user_registered`

Signal Arguments:

- **user:** The `settings.AUTH_USER_MODEL` instance that just registered.

This signal is triggered after a user has successfully registered. It will not be triggered if a user attempts to register with a duplicate email address.

Note: This signal fires before the user has verified their email address.

5.7 Contributing

We welcome contributions to this project. For non-trivial improvements, we would encourage discussion in a GitHub issue before opening a pull request. This is also a great contribution in and of itself as it lets us know where to focus development efforts.

5.7.1 Project Setup

The first step is to fork the repository and clone it to your computer. Once you have done that, we recommend setting up a virtual environment to encapsulate the project's dependencies. To install the third-party packages required by the project, run:

```
pip install -e .
```

Based on the contribution you are making, you may have to install additional dependencies.

```
# To run tests
pip install -r requirements/test.txt

# To build documentation locally
pip install -r requirements/docs.txt
```

Linting

We use black for code formatting and flake8 for linting. Code that fails the checks performed by either of these tools will cause the build to fail. To easily perform these checks on every commit, we recommend using the excellent `pre-commit` tool.

```
pip install pre-commit
pre-commit install
```

5.7.2 Testing

This project contains a comprehensive test suite. To run the tests locally, make sure the test requirements are installed and then execute:

```
pytest
```

5.7.3 Opening a Pull Request

Once you have made your contribution and pushed it to your fork of the repository, please open a pull request. The pull request description should contain a concise description of the change made and link to the issue that it addresses.

5.8 Changelog

5.8.1 v2.1.0

Features

- #71: Make email template paths configurable.

5.8.2 v2.0.3

Bugfixes * #77: Fix dependency issues for older versions of Django.

5.8.3 v2.0.2

Bugfixes

- #70: The email address a user registers with is now correctly marked as their primary email address.

5.8.4 v2.0.1

Bugfixes

- Fix clobbered version name.

5.8.5 v2.0.0

Breaking Changes

- Dropped support for Python versions less than 3.6 and added support for Python 3.7.

Features

- #59: Add setting to determine if users must provide their password when verifying an email address. To maintain backwards compatibility, the setting defaults to `True`.

Documentation

- #64: Add contributing guidelines.

Miscellaneous

- #61: Format code with `black`.

5.8.6 v1.2.0

Features

- Use `django-email-utils` for email sending. This allows us to easily send both HTML and plain text templated emails.

5.8.7 v1.1.0

Features

- #53: Emit a signal when an email address is verified.
- #54: Normalize email addresses.

5.8.8 v1.0.0

Features

- #47: Send a signal out when a user registers.

Bugfixes

- #42: Fix issue with creating multiple primary emails.
- #45: Confirmation tokens are now deleted once they have been used.
- #46: Documentation for endpoints using the generic `SerializerSaveView` is no longer broken.

Miscellaneous

- #41: Fix useless test.

5.8.9 v0.4.3

Bugfixes

- #44: Fix issue with templates not being included in distribution.

5.8.10 v0.4.2

Bugfixes

- #43: Fix issue with registration view not respecting overridden registration serializer setting.

5.8.11 v0.4.1

Bugfixes

- #40: Fix issue with invalid admin fields.

5.8.12 v0.4.0

Features

- #30: Add endpoints to request/perform a password reset.
- #37: Allow a custom registration serializer to be provided.

Documentation

- #29: Fix typo with installation instructions.

Miscellaneous

- #33: Fix issue with deployment process breaking example project requirements.

5.8.13 v0.3.1

Make dependency versions less strict.

5.8.14 v0.3.0

Features

- #9#25: Add documentation and example project.
- #10: Add custom authentication backend.
- #22: Add endpoints for managing email addresses.
- #24: Add field to track a user's primary email address.

5.8.15 v0.2.1

Bugfixes

- #20: Fix for tagged releases not being deployed.

5.8.16 v0.2

Features

- #4: Send a verification email after registration.
- #5: Add an endpoint for verifying email addresses.
- #6: Add an endpoint for resending an email verification.
- #7: Add a command for cleaning up expired email confirmations.

Miscellaneous

- #14: Email addresses must be unique

5.8.17 v0.1

Bare-bones intial release. This is not ready for any sort of use.

Features

- #2: Add endpoint to register new users.

5.9 Indices and tables

- genindex
- search

HTTP Routing Table

/emails

/register

/request-password-reset

/resend-verification

/reset-password

/verify-email