

---

# **django\_nopassword Documentation**

*Release 0.6.0*

**Rolf Erik Lekang**

**Sep 27, 2017**



---

## Contents

---

<b>1</b>	<b>Installation</b>	<b>1</b>
<b>2</b>	<b>Usage</b>	<b>3</b>
2.1	Verify users . . . . .	3
2.2	Custom backends . . . . .	4
<b>3</b>	<b>Settings</b>	<b>5</b>



# CHAPTER 1

---

## Installation

---

Run this command to install django-nopassword:

```
pip install django-nopassword
```

Requirements: Django >= 1.4 (1.5 custom user is supported)



Add the app to installed apps:

```
INSTALLED_APPS = (  
    'nopassword',  
)
```

Set the authentication backend to *EmailBackend*:

```
AUTHENTICATION_BACKENDS = ( 'nopassword.backends.email.EmailBackend', )
```

Add urls to your *urls.py*:

```
urlpatterns = patterns('',  
    url(r'^accounts/', include('nopassword.urls')),  
)
```

## Verify users

If it is necessary to verify that users still are active in another system. Override *verify\_user(user)* to implement your check. In *NoPasswordBackend* that method checks whether the user is active in the django app.

## Backends

There are several predefined backends. Usage of those backends are listed below.

**class** `nopassword.backends.email.EmailBackend`

Delivers the code by email. It uses the django send email functionality to send the emails. It will attach both HTML and plain-text versions of the email.

**class** `nopassword.backends.sms.TwilioBackend`

Delivers the code by sms sent through the twilio service.

## Custom backends

In `backends.py` there is a `NoPasswordBackend`, from which it is possible to build custom backends. The `EmailBackend` described above inherits from this backend. Creating your own backend is can be done by creating a subclass of `NoPasswordBackend` and implementing `send_login_code`. A good example is the `EmailBackend`:

```
class EmailBackend(NoPasswordBackend):

    def send_login_code(self, code, secure=False, host=None):
        subject = getattr(settings, 'NOPASSWORD_LOGIN_EMAIL_SUBJECT', _('Login code'))
        to_email = [code.user.email]
        from_email = getattr(settings, 'DEFAULT_FROM_EMAIL', 'root@example.com')

        context = {'url': code.login_url(secure=secure, host=host), 'code': code}
        text_content = render_to_string('registration/login_email.txt', context)
        html_content = render_to_string('registration/login_email.html', context)

        msg = EmailMultiAlternatives(subject, text_content, from_email, to_email)
        msg.attach_alternative(html_content, 'text/html')
        msg.send()
```



`django.conf.settings.NOPASSWORD_LOGIN_CODE_TIMEOUT`

Default: 900

Defines how long a login code is valid in seconds.

`django.conf.settings.NOPASSWORD_NAMESPACE`

Default: 'nopassword'

Defines the namespace for the urls, this must match the namespace of the include of `nopassword.urls`.

`django.conf.settings.NOPASSWORD_HIDE_USERNAME`

Default: False

If set to True, the login url will not contain the username.

`django.conf.settings.NOPASSWORD_LOGIN_EMAIL_SUBJECT`

Default: \_('Login code')

Sets Email Subject for Login Emails.

`django.conf.settings.NOPASSWORD_HASH_ALGORITHM`

Default: 'sha256'

Set the algorithm for used in logincode generation. Possible values are those who are supported in `hashlib`. The value should be set as the name of the attribute in `hashlib`. Example `hashlib.sha256()` would be `'NOPASSWORD_HASH_ALGORITHM = 'sha256'`.

`django.conf.settings.NOPASSWORD_POST_REDIRECT`

Default: True

By default, the login code url requires a POST request to authenticate the user. A GET request renders `registration/login_submit.html`, which contains some Javascript that automatically performs the POST on page load. To authenticate directly inside the initial GET request instead, set this to `False`.

`django.conf.settings.NOPASSWORD_CODE_LENGTH`

Default: 20

The length of the code used to log people in.

`django.conf.settings.NOPASSWORD_TWILIO_SID`  
Account ID for Twilio.

`django.conf.settings.NOPASSWORD_TWILIO_AUTH_TOKEN`  
Account secret for Twilio

`django.conf.settings.NOPASSWORD_NUMERIC_CODES`  
Default: False

A boolean flag if set to True, codes will contain numeric characters only (0-9).

`django.conf.settings.SERVER_URL`  
Default: 'example.com'

By default, `nopassword.views.login` passes the result of `result.get_host()` to `LoginCode.send_login_code` to build the login URL. If you write your own view and/or want to avoid this behavior by not passing a value for host, the `SERVER_URL` setting will be used instead.

`django.conf.settings.DEFAULT_FROM_EMAIL`  
Default: 'root@example.com'

## D

DEFAULT\_FROM\_EMAIL (in module django.conf.settings), 6

## E

EmailBackend (class in nopassword.backends.email), 3

## N

NOPASSWORD\_CODE\_LENGTH (in module django.conf.settings), 5

NOPASSWORD\_HASH\_ALGORITHM (in module django.conf.settings), 5

NOPASSWORD\_HIDE\_USERNAME (in module django.conf.settings), 5

NOPASSWORD\_LOGIN\_CODE\_TIMEOUT (in module django.conf.settings), 5

NOPASSWORD\_LOGIN\_EMAIL\_SUBJECT (in module django.conf.settings), 5

NOPASSWORD\_NAMESPACE (in module django.conf.settings), 5

NOPASSWORD\_NUMERIC\_CODES (in module django.conf.settings), 6

NOPASSWORD\_POST\_REDIRECT (in module django.conf.settings), 5

NOPASSWORD\_TWILIO\_AUTH\_TOKEN (in module django.conf.settings), 6

NOPASSWORD\_TWILIO\_SID (in module django.conf.settings), 5

## S

SERVER\_URL (in module django.conf.settings), 6

## T

TwilioBackend (class in nopassword.backends.sms), 3