Django Crypto Extensions

Release dev

Marczak

Jun 01, 2021

CONTENTS:

1 Requirements **1**

2 Installation **3**

3 Usage **5**
- 3.1 Django Models CryptoField Package 5
- 3.2 CryptoBinaryField and CryptoCharField 6

4 CONTRIBUTION **9**

5 Publishing new releases **11**

6 Indices and tables **13**

CHAPTER

ONE

REQUIREMENTS

Django Crypto Extensions requires a supported Django version and runs on Python versions 3.6 and above.

Refer to the project source code repository in GitHub and see the Tox configuration and Python package definition to check if your Django and Python version are supported.

CHAPTER

TWO

INSTALLATION

Django Crypto Extensions is easy to install from the PyPI package:

```
$ pip install django-crypto-extensions
```

After installing the package, the project settings need to be configured.

1. Add `django_crypto_extensions` to your `INSTALLED_APPS`:

```
INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'django.contrib.contenttypes',
    'django.contrib.sessions',
    'django.contrib.messages',
    'django.contrib.staticfiles',

    # Django Crypto Extensions app can be in any position in the INSTALLED_APPS list.
    'django_crypto_extensions',
]
```

CHAPTER

THREE

USAGE

3.1 Django Models CryptoField Package

A set of generic common Djano Fields that automatically encrypt data for database amd encrypt for the purpose of usage in Django:

- `CryptoFieldMixin` - Mixin that implement encrypt/decrypt methods.

Example of how to use to create custom CryptoTextField:

```
# models.py

from django.db import models

from django_crypto_extensions.django_fields import *

class CustomCryptoTextField(CryptoFieldMixin, models.TextField):
    pass
```

- `CryptoTextField` - TextField inheriting Field
- `CryptoCharField` - CharField inheriting Field
- `CryptoEmailField` - EmailField inheriting Field
- `CryptoIntegerField` - IntegerField inheriting Field
- `CryptoDateField` - DateField inheriting Field
- `CryptoDateTimeField` - DateTimeField inheriting Field
- `CryptoBigIntegerField` - BigIntegerField inheriting Field
- `CryptoPositiveIntegerField` - PositiveIntegerField inheriting Field
- `CryptoPositiveSmallIntegerField` - PositiveSmallIntegerField inheriting Field
- `CryptoSmallIntegerField` - SmallIntegerField inheriting Field

Settings

- each CryptoField has 2 kwargs `salt_settings_env` and `password`.
- salt_settings_env - name of variable stored in `settings.py` file, which will be used as cryptographic salt. `default: salt_settings_env = 'SECRET_KEY'` if settings not set or no SECRET_KEY in setting `default: salt = "Salt123!!!"`
- password - password to be used in encryption process of given field (together with salt set globally) `default = 'password'`

- password_field - name of field from which to import password. It i most recommended to ue @property instead of actual field

Example:

```
# models.py

from django.db import models

from django_crypto_extensions.django_fields import CryptoEmailField

class TestCryptoEmail(models.Model):

    value = CryptoEmailField(salt_settings_env='NEW_SECRET_KEY', password='new_password')
```

Example2:

```
# models.py

from django.db import models

from django_crypto_extensions.django_fields import CryptoEmailField

class CryptoTextModelPasswordFromField(models.Model):

    @property
    def password(self):
        return "password_field_to_be_used_as_key"

    value = CryptoTextField(password_field="password")
```

3.2 CryptoBinaryField and CryptoCharField

- A django-rest-framework fields for handling encryption through serialisation. Inputs are String object and internal python representation is Binary object for CryptoBinaryField and String object for CryptoCharField
- It takes the optional parameter `salt` (Django SECRET_KEY imported from setting as default). If set it use custom cryptographic salt
- It takes the optional parameter `password` ("Non_nobis1solum?nati!sumus" as default). If set it use a custom password in encryption. **It is highly recommended to use custom one!!**
- It takes the optional parameter `ttl` (None as default). If set it manage the number of seconds old a message may be for it to be valid. If the message is older than ttl seconds (from the time it was originally created) field will return None and encrypted message will not be enabled for decryption.

Example:

```
from rest_framework import serializers
from drf_extra_fields.crypto_fields import CryptoCharField


class CryptoSerializer(serializers.Serializer):
  crypto_char = CryptoCharField()
```

Example with parameters + It takes custom salt and password. Once saved it will be available for decryption for 1000 seconds.:

```
from rest_framework import serializers
from drf_extra_fields.crypto_fields import CryptoCharField


class CryptoSerializer(serializers.Serializer):
crypto_char = CryptoCharField(salt="custom salt", password="custom password", ttl=1000)
```

CHAPTER

FOUR

CONTRIBUTION

TESTS

Make sure that you add the test for contributed field to `test/*.py` and run with command before sending a `pull request`:

```
$ pip install tox  # if not already installed
$ tox
```

Or, if you prefer using Docker (recommended):

```
docker build -t django_crypto_extensions .
docker run -v $(pwd):/app -it django_crypto_extensions /bin/bash
tox
```

CHAPTER

FIVE

PUBLISHING NEW RELEASES

Increment version in `django_crypto_extensions/__init__.py`. For example:

```
__version__ = '0.2.2'  # from 0.2.1
```

Move to new version section all release notes in documentation.

Add date for release note section.

Replace in documentation all `New in Django Crypto Extensions development version` notes to `New in Django Crypto Extensions 0.2.2`.

Rebuild documentation.

Run tests.

Commit changes with message "Version 0.2.2"

Add new tag version for commit:

    $ git tag 0.2.2

Push to master with tags:

    $ git push origin main –tags

Don't forget to merge *master* to *gh-pages* branch and push to origin:

    $ git co gh-pages $ git merge –no-ff master $ git push origin gh-pages

Publish to pypi:

    $ python setup.py publish

CHAPTER

SIX

INDICES AND TABLES

- genindex
- modindex
- search