django-unfriendly Documentation

Release 0.3

Drew Engelson

February 01, 2015

Contents

The unfriendliest urls in town! `django-unfriendly` is a Django app that obfuscates urls and allows your application to natively execute the original view. There is lots of talk about SEO friendly urls. The trend is towards more and more human readable information in your urls and Django makes it easy to create urls like:

`http://yoursite.com/music/awesome/the-melvins/`

But sometimes urls can give too much away. This is where `django-unfriendly` comes in.

`django-unfriendly` provides a template filter that obfuscates urls in your templates, and a url handler/view that deobfuscates and executes the original view (e.g. no redirection).

CHAPTER 1

Why?

Perhaps you have a Django application with readable information in urls such as:

```
# original url
http://yoursite.com/music/awesome/the-melvins/
```

And you don't want nosy people trying to guess other urls:

```
# user guesses another url (that likely exists)
http://yoursite.com/music/sucks/the-cure/
```

You can obfuscate the url which might look like:

```
# obfuscated url
http://yoursite.com/u/E5v4uxuNSA8I2is33c6V8lqFTcdv_IxPLDGG/
```

Which will show the same response as the original url.

CHAPTER 2

Goals

1. The application must be completely transparent... Obfuscated urls should behave exactly like the original url. HTTP responses should be indistinguishable.
2. The original url must never be exposed (no redirect).
3. Tampering with the obfuscated url should be difficult. A tampered url should return a `404 - Page not found` error.
4. Obfuscated urls should be idempotent and may be safely cached. Or at least as cachable as the original url.

CHAPTER 3

Installation

1. Install the `django-unfriendly` package:

```
# with pip
pip install django-unfriendly

# or with easy_install
easy_install django-unfriendly
```

2. The only dependency is `pycrypto`. Make sure it's installed:

```
# with pip
pip install pycrypto

# or with easy_install
easy_install pycrypto
```

3. Add `unfriendly` to your `INSTALLED_APPS`:

```
INSTALLED_APPS = (
    ...
    'unfriendly',
    ...
)
```

4. Add `unfriendly.urls` to your `urls.py`:

```
urlpatterns = patterns('',
    ...
    url(r'^u/', include('unfriendly.urls')),
    ...
)
```

CHAPTER 4

Usage

Load the tag library into any templates where you want to use `django-unfriendly`:

```
{% load unfriendly_tags %}
```

Then apply the obfuscate filter to any url you'd like to hide:

```
<a href="{{ "/music/awesome/the-melvins/"|obfuscate }}">Melvins awesome</a>
```

Or with `{% url view %}` reversal:

```
{% url path.to.view as melvins_url %}
<a href="{{ melvins_url|obfuscate }}">Melvins awesome</a>
```

If SEO is still important to you, you can pass some SEO juice to the filter:

```
<a href="{{ melvins_url|obfuscate:"King Buzzo rocks" }}">Melvins awesome</a>
```

CHAPTER 5

Settings

The following may be added to your setting.py to customize the behavior of this app.

- `UNFRIENDLY_SECRET`
  - default: `SECRET_KEY` (well, first 32 bytes of it)
  - Random key used for encryption/decryption. Note: AES keys must be either 16, 24, or 32 bytes long.
- `UNFRIENDLY_IV`
  - default: `SECRET_KEY` (well, first 16 bytes of it)
  - Initial vector required by AES cipher. Note: AES initial vector must be 16 bytes long
- `UNFRIENDLY_ENFORCE_CHECKSUM`
  - default: `True`
  - Determines whether or not the decrypted data is validated against a crc checksum to detect tampering.