Django Featurette Documentation

Release 0.1

Germano Guerrini

August 31, 2015

Contents

Django Featurette is a simple application that helps your website to offer certain feature only to selected users.

Suppose for example that you have one or more groups of *premium* users, and that a given page is only visible to a certain group, or that a section of an otherwise public page should be rendered only for another group.

There are three ways to achieve that goals with Featurette:

- Using a decorator on a view
- Wrapping within a template tag a section of your template
- Fine-grained control through helper methods

We are going to show all these methods in the Usage guide.

CHAPTER 1

Requirements

Python 2	>= 2.7
Python 3	>= 3.2
Django	>= 1.6

CHAPTER 2

Contents

2.1 Getting Started

2.1.1 Installation

Use your favorite Python package manager to install the app from PyPI, e.g.

Example:

```
pip install django_featurette
```

2.1.2 Configuration

Add `django_featurette` to the `INSTALLED_APPS` within your settings file (usually `settings.py`).

Example:

```
INSTALLED_APPS = [
    [...]
    'django_featurette',
]
```

After that, run the `syncdb` command, or `migrate` if you are using Django 1.7:

```
./manage.py syncdb
```

As Featurette rely on built-in Django authentication system, you will need to add all the needed applications and middleware. Head to the official documentation if you need help.

2.2 Usage

The first thing to do is to create one instance of `Feature` for each feature you want to implement.

Basically, a feature is simply an arbitrary key that can be associated to a *Group* of users (the way Django intends it) and is active over a certain time window.

To create a new feature just use the supplied admin interface. The same goes for the group and the relative users, given that you installed the admin provided by Django.

Now, let's say we have an active feature identified by the key *premium_feature*. Let's examine the three way you can use that to show some given content only to users within the group associated to the feature.

2.2.1 Decorate a view

If that's what your feature requires, you can return an *HttpResponseForbidden* instead of the normal response of a view using a decorator:

```
from django_featurette.decorators import user_enabled

@user_enabled('premium_feature')
def premium_view(request):
    [your code goes here]
```

2.2.2 Using a template tag

If you just need to hide some content from a template, you can use a template tag:

```
{% load featurette %}

{% feature premium_feature %}
You can see this because we love you
{% endfeature %}
```

2.2.3 Fine-grained control

As a last resource, you can use an helper method to control your view flow (or other methods that handles users):

```
from django_featurette.utils import is_feature_enabled_for_user

def get_premium(user):
    gift = '1 dollar'
    if is_feature_enabled_for_user('premium_feature', user):
        gift = '10 dollars'
    return gift
```

That's all!

2.3 API Reference

2.3.1 Classes

2.3.2 Utilities

2.3.3 Templatetags

2.4 Changelog

2.4.1 Version 0.4

- Added test support for Django 1.8

2.4.2 Version 0.3

- Added migrations files for Django 1.7+

2.4.3 Version 0.2

- Supports for Python 3

2.4.4 Version 0.1

- First stable version

CHAPTER 3

Indices and tables

- genindex
- search