tastytools Documentation

Release 0.1.1

Ignacio Munizaga, Juan Enrique Muñoz, Sebastian Acuña

May 30, 2017

Contents

Tastytools is a set for usefull tools to develop a quality tastypie webservice API.

It's main features are automatic documentation and the generation of Hygiene tests (tests that ensure the pressence of certain features that that do not give positive satisfaction, though dissatisfaction results from their absence). For example it tests the pressence of help fields An example in the case of an API, is a help text on the fields

CHAPTER 1

Getting Started with TastyTools

For example purposes, we'll be adding tools to the simple blog application that Tasytypie has in it's own tutorial.

Here is the code we'll be using (taken from the tastypie quickstart and tastypie tutorial with a few minor changes).

`myapp/models.py`:

```
from django.contrib.auth.models import User
from django.db import models
from django.template.defaultfilters import slugify


class Entry(models.Model):
    user = models.ForeignKey(User)
    pub_date = models.DateTimeField(auto_now_add=True)
    title = models.CharField(max_length=200)
    slug = models.SlugField()
    body = models.TextField()

    def __unicode__(self):
        return self.title

    def save(self, *args, **kwargs):
        # For automatic slug generation.
        if not self.slug:
            self.slug = slugify(self.title)[:50]

        return super(Entry, self).save(*args, **kwargs)
```

`urls.py`:

```
from django.conf.urls.defaults import patterns, include, url
from django.contrib import admin
from tastypie.api import Api
from myapp.api.resources import EntryResource, UserResource

v1_api = Api(api_name='v1')
```

```
v1_api.register(UserResource())
v1_api.register(EntryResource())

admin.autodiscover()

urlpatterns = patterns('',
    url(r'^admin/', include(admin.site.urls)),
    (r'^api/', include(v1_api.urls)),
)
```

myapp/api/resources.py:

```
from django.contrib.auth.models import User
from tastypie import fields
from tastypie.resources import ModelResource
from myapp.models import Entry


class UserResource(ModelResource):
    class Meta:
        queryset = User.objects.all()
        resource_name = 'user'


class EntryResource(ModelResource):
    user = fields.ForeignKey(UserResource, 'user')

    class Meta:
        queryset = Entry.objects.all()
        resource_name = 'entry'
```

You can download this project by cloning git@github.com:thagat/django_tastyblog.git

Installation

Simply clone the repository:

```
git clone https://github.com/juanique/django-tastytools.git
cd django-tastytools
python setup.py install
```

Configuration

Add `'tastytools'` to your `INSTALLED_APPS`

Generating documentation

For our api to be easily consumable by users, we need documentation. Tastytools generates automatic documentation, so your clients always have the latest api docs. For our simple application, we'll create a file: `myapp/api/tools.py` (in the api folder created within your app in the tastypie quickstart).

First thing we need to do is move much of the tastypie code form the urls.py file to the new tools.py file:

```
# myapp/api/tools.py
from tastytools.api import Api
from myapp.api.resources import EntryResource, UserResource

v1_api = Api(api_name='v1')
v1_api.register(EntryResource())
v1_api.register(UserResource())
```

Notice that our Api object does not import form tastipie anymore, we'll be using from now on the tastytools Api class, that inherits from the tastypie Api class.

Our urls.py file now needs to import the api object to keep working, and we'll add a new line to generate our documentation:

```
# urls.py
# ...
from myapp.api.tools import v1_api

urlpatterns = patterns('',
    # ...
    (r'^api/', include(v1_api.urls)),
    # Then add:
    (r'^tastytools/', include('tastytools.urls'), {'api_name': v1_api.api_name}),
)
```

Now you can go check your auto generated documentation at /tastytools/doc/ Neat right? it's now easy to navigate through your api resources.

Generating Example Data for your Tastypie API

Every great documentation has examples, so tastytools helps you with this by generating example data: Implement a Test Data class, it's the one in charge of creating data for our tests:

```
# myapp/api/tools.py
from tastytools.test.resources import ResourceTestData


class EntryTestData(ResourceTestData):
    resoruce = "entry"

    def get_data(self, data):
        data.set('user', resource='user')
        data.set('pub_date', '2010-12-24T06:23:48')
        data.set('title', 'Lorem ipsum')
        data.set('slug', 'lorem')
        data.set('body', 'Lorem ipsum ad his scripta blandit partiendo...')
        return data


class UserTestData(ResourceTestData):
    resource = "user"

    def get_data(self, data):
    data.set('username', 'foo')
```

```
    data.set('email', 'bar@foo.com')
    return data
```

Then register our test data to our api:

```
v1_api.register_testdata(EntryTestData)
```

Generating Tests for your Tastypie API

The second great feature of tastytools is that it can generate a number of tests for your api. This tests seek to ensure among other things, the readability of your api:

```
#myapp/api/tests.py
from tastytools.test.definitions import resources, fields
from api.tools import v1_api

ResourceTests = resources.generate(v1_api)
ResourceFieldTests = fields.generate(v1_api)
```

Remember to add this test.py file to the set of tests your application tests by importing it to your tests.py file or your tests/__init__.py file if you have your tests in a folder

Note: For the tests to work you need to register TestData classes to the api object

Now you have a lot of new tests for your api, which you can run with the ./manage.py tests myapp command. Fix them and your api will gain more than a level in usability :D.

CHAPTER 2

Quick Start

Assuming you have a tastypie api and have already read the tastypie docs:

1. Add `tastytools` to `INSTALLED_APPS`.
2. Create an file in `<my_app>/api/tools.py`, and move the tastypi api definition from the urls.py file to it:

```
from tastytools.api import Api
from <my_app>.api.resources import MyModelResource
from <my_app>.api.resources import AnoterResource, YetAnotherResource

api = Api()
api.register(MyModelResource)
api.register(resources=[AnotherResource, YetAnotherResource])
```

3. Then import the api to your urls root file:

```
from my_app.api.tools import api

urlpatterns = patterns('',
  # ...more URLconf bits here...
  (r'^api/', include(api.urls)),
  # Then add:
  (r'^tastytools/', include('tastytools.urls'), {'api_name': api.api_name}),
)
```

4. got to http://localhost:8000/tastytools/v1/.

As you can see, now you have documentation for anyone who wants to consume your api resources!

CHAPTER 3

Requirements

Tastytools requires Tastypie to work. If you use Pip, you can install the necessary bits via the included `requirements.txt`:

- django-tastypie (http://django-tastypie.readthedocs.org/)