Django Kong Documentation

Versión 0.9

Eric Holscher

27 de julio de 2017

Índice general

1. A simple example **1**

2. Get the code **3**

3. Contents **5**

3.1. Installation 5
3.2. Management Commands 6
3.3. Settings 6
3.4. Overview 7
3.5. Meta Documentation 10
3.6. Roadmap 10

4. What's with the name? **13**

CAPÍTULO 1

A simple example

You can see a basic version running for my personal site. It is super barebones, but it should give you an idea of what exactly is possible.

Tests are written using Twill, which allows for easy functional testing of web apps.

CAPÍTULO 2

Get the code

The source is available on Github. I would like to thank Nathan Borror for the design parts that are pretty :)

The mailing list for the project is located at google groups: http://groups.google.com/group/django-testing

CAPÍTULO 3

Contents

Installation

Installing Kong is pretty simple. Here is a step by step plan on how to do it.

Nota: Kong is available on Pypi as `django-kong`, but trunk is probably your best best for the most up to date features.

First, obtain Python and virtualenv if you do not already have them. Using a virtual environment will make the installation easier, and will help to avoid clutter in your system-wide libraries. You will also need Git in order to clone the repository.

Once you have these, create a virtual environment somewhere on your disk, then activate it:

```
virtualenv kong
cd kong
source bin/activate
```

Kong ships with an example project that should get you up and running quickly. To actually get kong running, do the following:

```
git clone http://github.com/ericholscher/django-kong.git
cd django-kong
pip install -r requirements.txt
pip install . #Install Kong
cd example_project
./manage.py syncdb --noinput
./manage.py loaddata test_data
./manage.py runserver
```

This will give you a locally running instance with a couple of example sites and an example test.

Now that you have your tests in your database, you need to check that your tests run. You can run tests like:

```
#Check all sites
./manage.py check_sites
#Only run the front page test
./manage.py check_sites -t front-page
#Only check sites of type Mine
./manage.py check_sites -T mine
```

The first command is the default way of running kong, and will run the tests for all of your sites.

The second two different ways will run either a specific test, or a type of test. Both of these can run tests across multiple sites.

Management Commands

Kong ships with management commands that allows you to easily run your tests. There are a couple of different ways to run tests, based on how you want things to work.

check_sites

By default, this runs tests for all your sites

This is the main command that is used to run kong tests. By default, it does nothing, but you can pass in arguments to it, to make it do what you want.

Arguments

-t, –test <test_slug>

Run the tests for a specific `Test`. This might be run across multiple ‘‘Site‘‘s or ‘‘Type‘‘s, whatever the test defines.

-s, –site <site_slug>

Run all of the tests for a specific site. This will run the tests that explicitly point at the `Site`, as well as the tests for the `Type` that the site is.

-T, –type <type_slug>

Run the tests for a specific `Type`. This will run the tests for all sites for that `Type`.

Settings

Kong has a number of settings that will effect the behavior of Kong. Mostly related to the sending of notifications.

KONG_MAIL_MANAGERS

Default: False

When set to true, this mails notifications to the people defined in Django’s MANAGERS setting.

KONG_MAIL_ADMINS

Default: False

Like KONG_MAIL_MANAGERS, when set to `True`, this mails notifications to the people defined in Django's ADMINS setting.

KONG_MAIL_ON_RECOVERY

Default: True

When `True`, you are notified when your test has been fixed, as well as when it breaks.

KONG_MAIL_ON_EVERY_FAILURE

Default: False

When `True`, this will send you an email on every test that fails. If `False`, you will only get emails on the first time that this test fails. This would presumably be used along with KONG_MAIL_ON_RECOVERY, so that you only get mails when a test fails and then passes again.

KONG_MAIL_ON_CONSECUTIVE_FAILURES

Default: 1

When set to a value above `1`, only send emails when a test has failed x number of times.

KONG_RESET_BROWSER

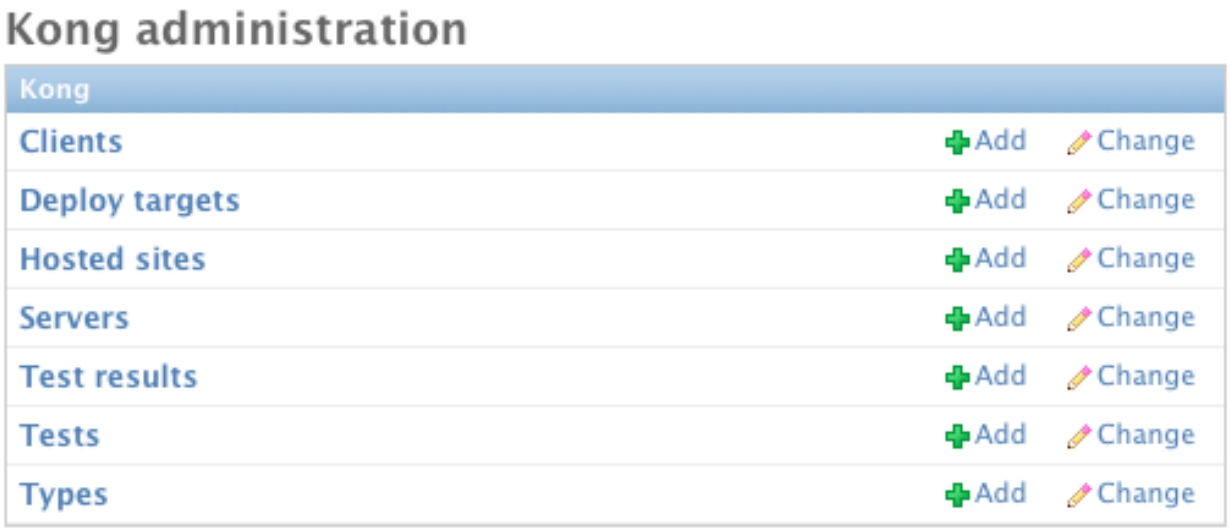
Default: False

When set to `True`, the browser is reset between tests. This means in essence that all cookies are reset.

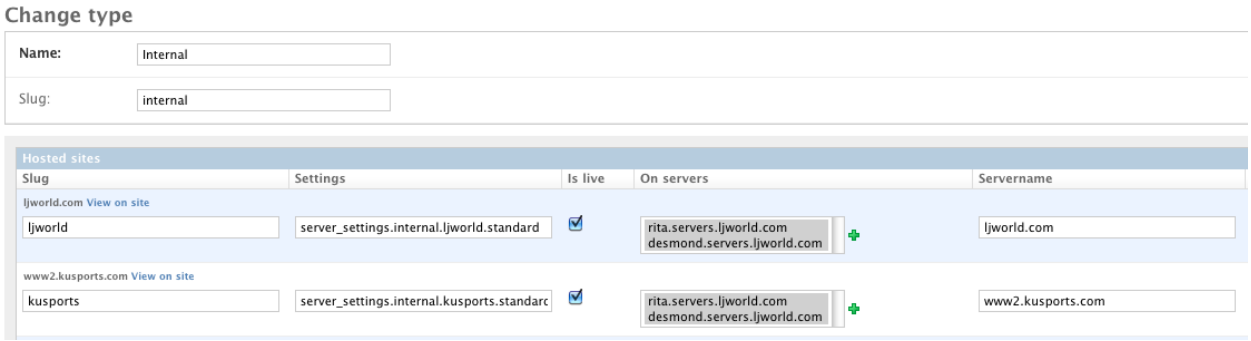
Overview

Getting Started

At work we have to manage a ton of Django based sites. Just for our World Company sites, we have over 50 different settings files, and this doesn't take into account the sites that we host for other clients. At this size it becomes basically impossible to test each site in a browser when you push things to production. To solve this problem I have written a very basic server description tool. This allows you to describe sites (settings file, python path, url, etc.) and servers.



On top of this base, I have written a way to run tests against these sites. You can categorize the sites by the type of site they are (We have Marketplace, ported Ellington, and old Ellington sites). This allows you to run tests against different types of sites. You may also have custom applications that run on only one or two certain domains. You can specify specific sites for tests to be run against as well.

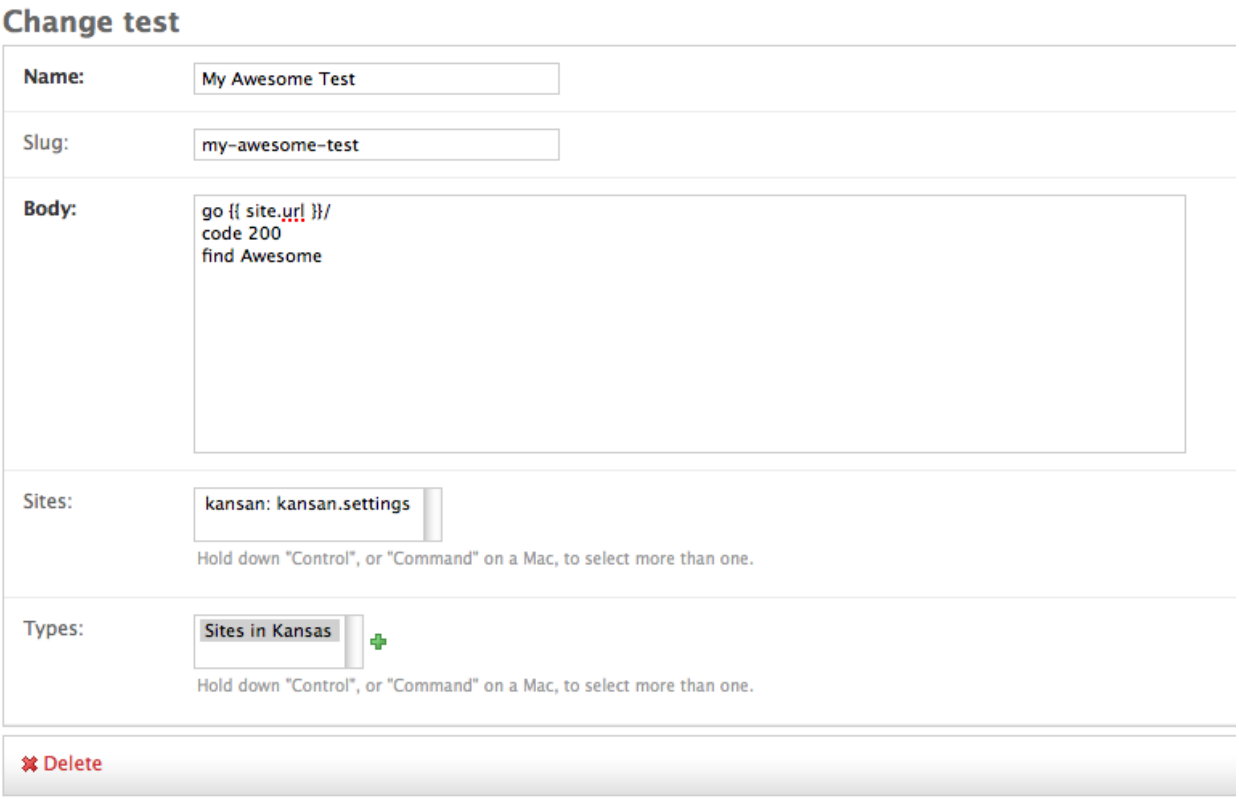


The tests are written in Twill, which is a simple Python DSL for testing. Twill was chosen because it is really simple, and does functional testing well. The twill tests are actually rendered as Django templates, so you get the site that you are testing against in the context. A simple example that tests the front page of a site is as follows:

```
go {{ site.url }}
code 200
find "Latest News"
```

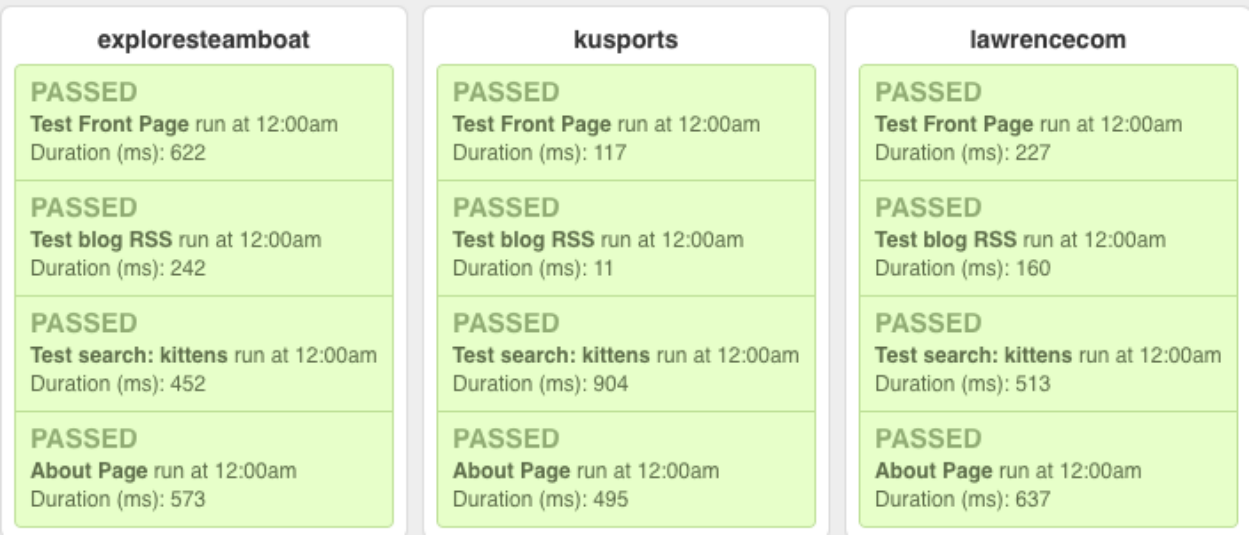
This simply loads the Site's front page, checks that the status code was 200, and checks that the string Latest News is on that page. The arguments to find are actually a regex, allowing for lots of power in checking for content.

The interface for this in the Admin is pretty simple.



You can see that this Test will run against any of the Sites that we have defined in the "Sites in Kansas" Type.

This then gives you the ability to view all of the results for your tests in a web interface. Below is an example of the live view that I see when looking at our servers. We have only just started using Kong, but the tests it provides are really useful to make sure that functionality works after a deployment.



You can also see the history of a test on a site. Currently it shows the last 15 results, but paginating this page will be easy. It allows you to see if your test has been running well over time. Another nice thing is that it measures the Duration of the test, so that you can see if it is going slow or fast.

As you can see, the data display is really basic. It will be improved, but currently its basically the "simplest thing that

could possibly work".

Using it yourself

When we deploy code changes, I generally run the Kong tests against our sites, making sure that things work. When we launch something new, I will write a kong test to exercise it across all sites. The tests usually take a minute to write, and save lots of time and heart ache, knowing all the sites work.

At the moment the tests can be kicked off by a django management command. The *check_sites* command will allow you to run all of the tests for a given Type or Test. Allowing you to run all of the Ellington tests across all sites, or just run one test across all sites.

We currently have this wired up to a cron job that runs every 10 minutes. If you set the *KONG_MAIL_MANAGERS* settings to True, it will send an email to the site managers on a test failure. At some point in the future, I will be integrating Kong into Nagios, so that Nagios will handle the running and alerting of errors. That is eventually the way that it will be run.

There are a lot of ways that this can be improved, however in it's current state it works for me. I figured releasing it will allow anyone who needs something like this to be able to use it. There is no documentation or tests, which will be fixed soon! The web display can also be improved a ton, and that is a high priority as well.

Meta Documentation

What is the point

Kong came about to solve a problem. At the Lawrence Journal-World, we have over 20 sites that we maintain that run a couple different versions of software that we make. Every time we wanted to push code live, we had no good way of making sure that we didn't break shit, other than hand testing sites or spidering them. Kong is a middle ground in between those 2 approaches, allowing you to specify certain behaviors that you want to test across all of your sites. By using Twill as the language, it lets us do interesting things like fill out forms and follow links, providing you with interesting ways of testing that your sites are functioning correctly after a deployment.

Basic Architecture

Kong has two main top-level ideas. *Site's and 'Types*. A *Type* is a collection of 'Site's. When you define a test, you either define it for a Type of site – which will then apply to every Site in that Type. You can also assign a test to a specific Site, for something that is custom functionality for that site. The idea is to define tests mostly on Type's, which will then automatically be run against all sites added to that Type. Then for special cased functionality, you can run that on only a collection of 1 or 2 sites.

Roadmap

1.0

For 1.0 I want to make Kong kick ass at running Twill tests in the browser. This is it's main purpose, and 1.0 will hopefully hit this out of the park.

2.0

In 2.0 I want to expand Kong to more than Twill tests. I want to be able to run twill tests off of the filesystem. Also I want to be able to have other "backend" types of tests, basically anything returning a 0/1/2, much like nagios.

CAPÍTULO 4

What's with the name?

Originally Kong was called paradigm, because it was going to change the way we thought about deployment. After much convincing from coworkers that this was too enterprisey, during the Djangocon 09 sprints, I was given the name Donkey Kong. I thought it would be a fun play on words to name a project django kong, because it sounds like Djangocon, and it plays off of Donkey Kong. Then I just needed to find a way to associate Kong with what the project actually does, because it's a Deployment Testing Tool for King/Donkey Kong sized sites.