Django Kong Documentation

Release 0.9

Eric Holscher

Mar 06, 2018

Contents

1 A simple example **1**

2 Get the code **3**

3 Contents **5**

3.1 Meta Documentation 5

3.2 Tutorial 5

4 What’s with the name? **9**

CHAPTER 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.

CHAPTER 2

Get the code

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

CHAPTER 3

Contents

3.1 Meta Documentation

3.1.1 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.

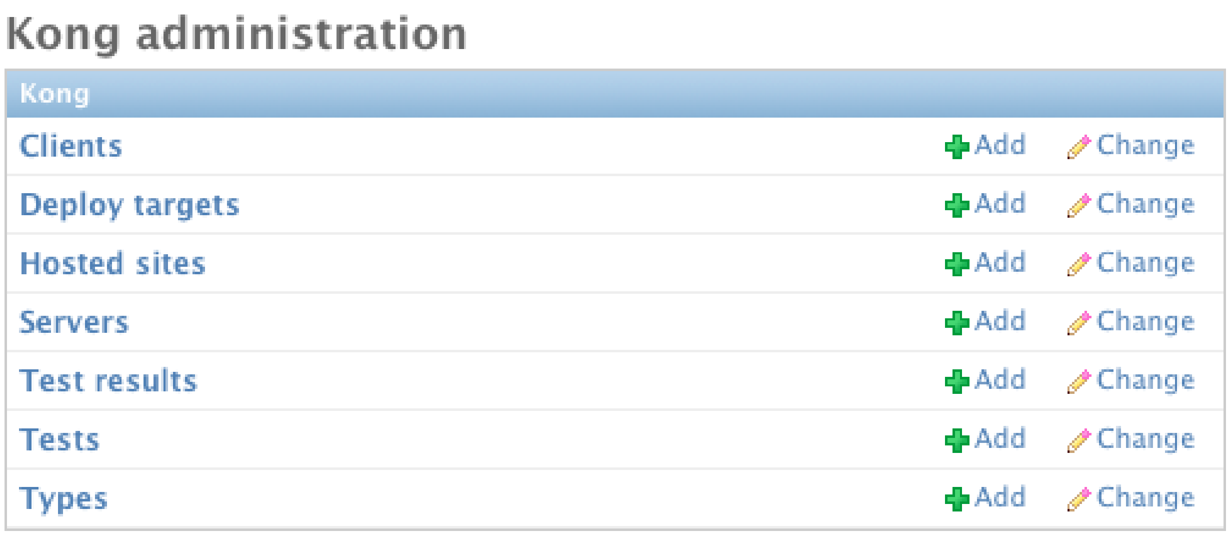
3.1.2 Basic Architecture

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.

3.2 Tutorial

3.2.1 Getting Started

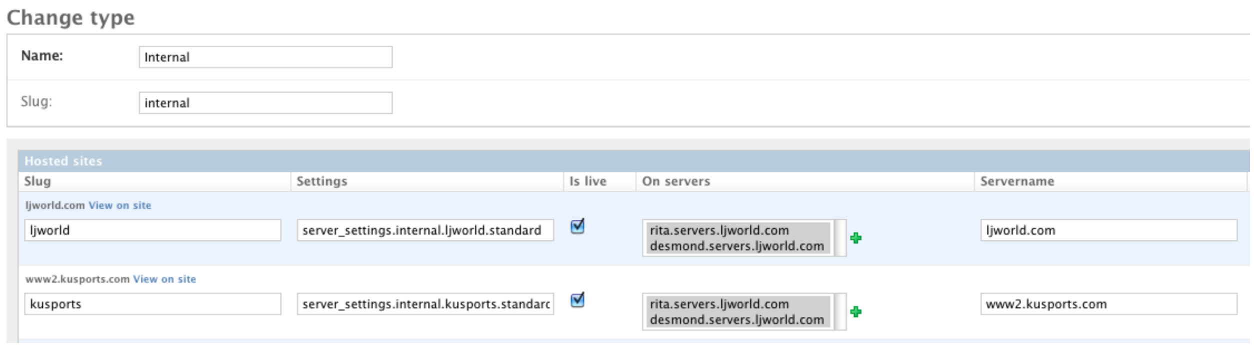
At work we have to manage a ton of Django based sites. Just for our World Company sites, we have over 50 different settings files, and this doesn't take into account the sites that we host for other clients. At this size it becomes basically impossible to test each site in a browser when you push things 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.



Kong administration

Kong		
Clients	➕Add	Change
Deploy targets	➕Add	Change
Hosted sites	➕Add	Change
Servers	➕Add	Change
Test results	➕Add	Change
Tests	➕Add	Change
Types	➕Add	Change

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.



Change type

Name:	Internal
Slug:	internal

Hosted sites

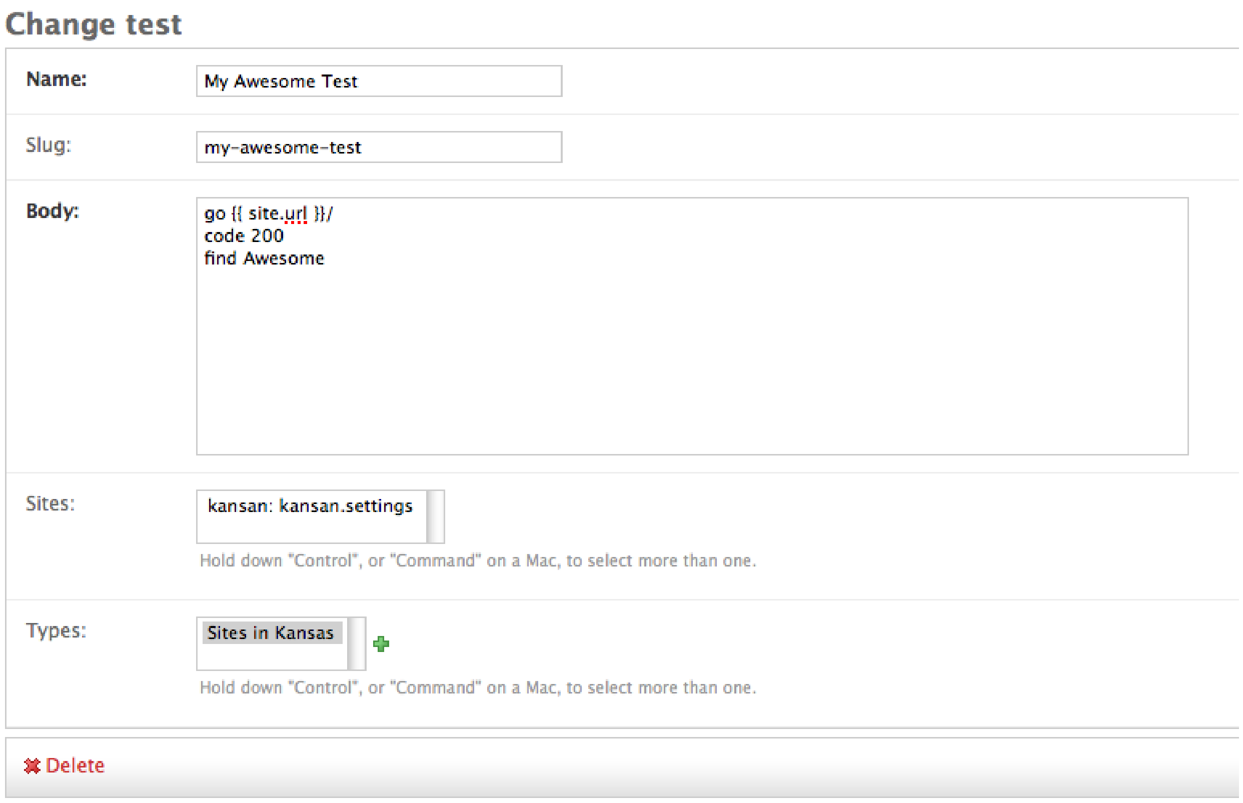
Slug	Settings	Is live	On servers	Servername
ljworld.com View on site				
ljworld	server_settings.internal.ljworld.standard	✓	rita.servers.ljworld.com desmond.servers.ljworld.com ➕	ljworld.com
www2.kusports.com View on site				
kusports	server_settings.internal.kusports.standarc	✓	rita.servers.ljworld.com desmond.servers.ljworld.com ➕	www2.kusports.com

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.



Change test

Name: My Awesome Test

Slug: my-awesome-test

Body:

```
go {{ site.url }}/
code 200
find Awesome
```

Sites: kansan: kansan.settings

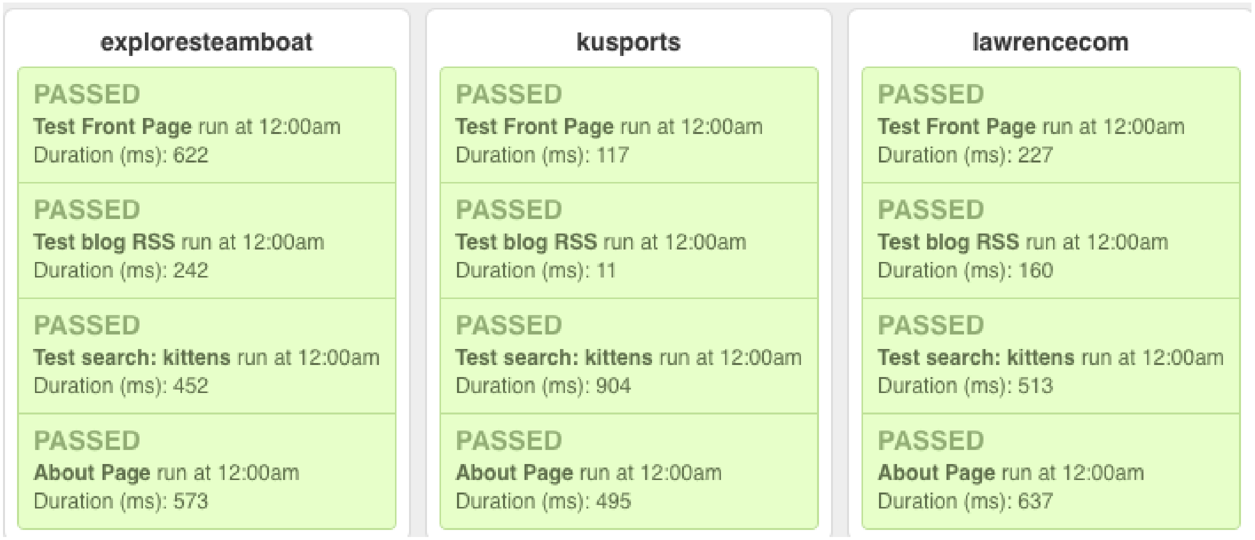
Hold down "Control", or "Command" on a Mac, to select more than one.

Types: Sites in Kansas

Hold down "Control", or "Command" on a Mac, to select more than one.

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.



explorensteamboat

PASSED
Test Front Page run at 12:00am
Duration (ms): 622

PASSED
Test blog RSS run at 12:00am
Duration (ms): 242

PASSED
Test search: kittens run at 12:00am
Duration (ms): 452

PASSED
About Page run at 12:00am
Duration (ms): 573

kusports

PASSED
Test Front Page run at 12:00am
Duration (ms): 117

PASSED
Test blog RSS run at 12:00am
Duration (ms): 11

PASSED
Test search: kittens run at 12:00am
Duration (ms): 904

PASSED
About Page run at 12:00am
Duration (ms): 495

lawrencecom

PASSED
Test Front Page run at 12:00am
Duration (ms): 227

PASSED
Test blog RSS run at 12:00am
Duration (ms): 160

PASSED
Test search: kittens run at 12:00am
Duration (ms): 513

PASSED
About Page run at 12:00am
Duration (ms): 637

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".

3.2.2 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.

CHAPTER 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 :)