slug.in Documentation

Release 1.0

Chris Dickinson, Wraithan, Eric Holscher

March 18, 2013

CONTENTS

We have built a generic, learning URL shortner. AKA, django.me on steroids.

Because it is the Django Dash, we felt the need to overarchitect it in just about every way. On our vision quest, we have learned many things about the inner working of Tastypie, Redis, and the World.

Note: it turns out nosql databases aren't great at relational data

who knew

CHAPTER

ONE

WHAT WE BUILT

- **A generic Tastypie API Admin interface “Pecan”**
    - You can point it at any Tastypie API, and get basic CRUD
    - Understands what tastypie methods are allowed
    - Visible at http://slug.in/_admin/
    - **You’ll need to login to use it**
        - This is protected by the web-scale-iest of security methods: *Authorization: Basic <base64 yourname:yourpassword>*
- **A Tastypie API for a Redis data store**
    - This exposes basic CRUD operations for a couple “Models”
    - It works in conjunction with the data store
    - Has the basic underpinnings for a generic RedisResource for Tastypie, still needs a ton of work tho.
    - Visible at http://slug.in/_api/v1/?format=json
- **Analytics gathering and API**
    - Provides a Tastypie API on top of Postgres
    - Works with Pecan
    - Tracks what redirects are followed, and used the most
- **An importer for Intersphinx data**
    - Currently we only index data from Sphinx
    - We can index any arbitrary intersphinx page, so people can host this themselves
- **The main site**
    - Provides a instant-filtering interface for finding shortcuts
    - Allows users to add new links
    - Uses Pecan to provide editing functionality
    - Tracks what links are clicked the most, so you can crowd-source the best shortcuts
    - Visible at http://slug.in/

CHAPTER

TWO

API

2.1 Redis Data Model

```
"hydra:v1:projects" = Index of projects <Set>
"hydra:v1:projects:<project>" = Project Metadata <Hash>
"hydra:v1:projects:<project>:slugs" = Index of slugs <Set>
"hydra:v1:projects:<project>:slugs:<slug>" = Slug Data <SortedSet>
```

2.2 REST API

```
# List of projects
GET /_api/v1/projects/?format=json
{
    "meta": {},
    "objects": [
        {
            "name": "readthedocs",
            "resource_uri": "/_api/v1/project/readthedocs/",
            "whitelist": [
                "djangoproject.com",
                "readthedocs.org"
            ]
        },
        {
            "name": "fabric",
            "resource_uri": "/_api/v1/project/fabric/",
            "whitelist": [
                "djangoproject.com",
                "fabfile.org"
            ]
        }
    ]
}

# Project Detail
GET /_api/v1/projects/<project>/
{
    "name": "django",
    "resource_uri": "/_api/v1/project/django/",
    "whitelist": [
        "djangoproject.com",
        "readthedocs.org"
    ]
}

# List of Redirects
GET /_api/v1/redirect/?format=json
{
    "meta": {},
    "objects": [
        {
            "project": "django",
            "resource_uri": "/_api/v1/redirect/django/get_object/",
            "slug": "get_object",
            "urls": [
                {
                    "score": 1.0,
                    "url": "http://django.readthedocs.org/en/latest/ref/class-based-views/mixins-sing
                }
            ]
        }
    ]
}

# Redirect detail
GET /_api/v1/redirect/django/get_object/?format=json
{
    "project": "django",
    "resource_uri": "/_api/v1/redirect/django/get_object/",
    "slug": "get_object",
    "urls": [
        {
            "score": 1.0,
            "url": "http://django.readthedocs.org/en/latest/ref/class-based-views/mixins-single-objec
        }
    ]
}

#Delete project
DELETE /_api/v1/project/django/

#Delete redirect
DELETE /_api/v1/redirect/django/get_object/

#Edit project
PUT /_api/v1/project/django

#Edit redirect
PUT /_api/v1/redirect/django/get_object/
```