Singkat: A Keyword-Based URL Shortener and Click Tracker Package for Django Web Application

Gottfried Prasetyadi, Utomo Tri Hantoro, Achmad Benny Mutiara
Faculty of Computer Science and Information Technology
Gunadarma University
Depok, Indonesia

***Abstract*—In recent years, Python has been gaining popularity as web scripting/programming language. In this research we propose Singkat, an open source uniform resource locator (URL) shortener package with web user interface built using Python-based Django web framework. It can be deployed and customized in any web project based on Django framework. This makes sure that administrators can gain control over data in their own environment. Users can create shortened links using base62 values to generate pseudo random keyword. To minimize phishing and other abuses, only registered users can create shortened link using their chosen keyword, and it is possible to preview a link before accessing it. Registered users can also monitor each click and get useful information. We also ran some tests to measure Singkat's performance and functionality.**

***Keywords*—Url shortener; click tracker; python; django framework; open source**

I. Introduction

A web address is a string used to identify a web resource. It includes uniform resource locator (URL) and other strings of characters, which can be used to identify web resources when processed appropriately [1]. Internet today is full of content and each web page needs its own unique URL. To make a URL identifiable by human, some organization must be applied to make it more structured e.g. by using slashes, date of content creation, slugs and keywords from article and blog posts [2]. However, this makes URL hard to memorize, type manually, or distribute (especially via non-digital media), and can only be copied-pasted for reliability.

URL shortener can shorten long URL into shortened one. This kind of web service has been available for at least 16 years [3]. Tinyurl.com and bit.ly are the examples of popular closed source, non-free URL shortener services for public consumption. There are also open source web modules and packages such as Polr, YOURLS, and Shlink, all of which can be used as extension in a web application within a domain. Some are available to Python-based web application, such as microurl, djanurl, and lilliput.

Python is one of the fastest-growing major programming language. It has risen in the ranks, surpassing both C# and PHP in 2018 [4, 5]. There are many web framework based on Python, such as Django, Web2py, and Flask microframework. According to [4], 13% of programmers worldwide use Django framework in 2018. It ranked sixth in the most commonly used frameworks and libraries. Django employs model-view-template (MVT) architectural pattern, which enables code reuse and modularity of features and functions [6].

Singkat is aimed to be a highly customizable URL shortener module alternative for web applications built using Django, with complete user features for a basic deployable package, such as web user interface, shortening using pseudo random and user-chosen keyword, and clicks statistics. The name 'Singkat' is taken from Indonesian, which means short, brief, or concise. This research centered on algorithm and performance of Singkat, especially its model object, harnessing Python scripting language.

II. Research Methods

A. Previous Works

There are several URL shortener modules for Django web framework similar to Singkat. Compared to them, Singkat has distinct differences. Microurl (pypi.org/project/microurl) is Python library for URL minification, which requires external services for authentication: Google or bitly. Djanurl (github.com/ericls/djanurl) is a URL shortener module written in Python version 3. Lilliput (github.com/Feverup/lilliput) is a URL shortener based on older version of Django. It treats links as model objects; each has integer identifier converted to base62 characters, with a fixed length of four characters. All of them are open source and can be found on online code repositories.

B. Models and Relationships

We defined four core models: 'singkat', 'clicker', 'click', and 'clickdetail'. As in any other MVT framework, a model is a table in a database.

Singkat model is used to represent shortened URLs. Clicker model is used to store data of clients that accessed a singkat object. Click model is used as intermediate model to facilitate many-to-many relationship of singkat model and clicker model. Clickdetail model is used to store timestamp of every clicks. The Django built-in user model is also used to represent registered users of the system.

Table 1 to 4 show the detail of attributes of each core model: the type, length, and other options, e.g., a model should have one unique attribute as primary key. Fig. 1 shows the model relationship; it can be concluded that a 'click' model is used to resolve the many-to-many relationship between 'singkat' and 'clicker' models, which has one-to-many relationship with 'clickdetail' model.

C. Algorithm for Creating Singkat Object

For both pseudo random and user-chosen keywords, we used 'requests' [7], a Python HTTP library, to check whether the input long URL is accessible at that time, preventing link to invalid URL [8], in contrast to Djanurl. By default, maximum user-chosen keyword length is 100 characters, while maximum long URL input length is 2000 characters to meet the specifications of RFC 7230 [9] and popular web browsers.

1) Pseudo Random Keyword

The pseudocode for creating singkat object with pseudo random keyword is as follows:

```
1. If target_url is not valid: raise error 'Cannot
   access target URL'
2. If target_url is singkat: raise error 'Already a
   Singkat URL'
3. new_singkat = Singkat(target_url,
   keyword=generate_random_keyword())
```

Every new Singkat object is instantiated from Singkat class, which takes two main parameters: the target URL and the keyword, which in this case is generated automatically using generate_random_keyword() function.

We used pseudo random keyword identifier (represented as integer), which is designed to be available globally in the system, and converted it to base62 alphabet to generate pseudo random keyword, so that the newly generated keyword would always be available and not already in use. The alphabet consists of 0-9, A-Z, and a-z, so there are $\sum_{n=1}^{100} 62^n$ keyword possibilities, which is different from Lilliput. The generate_random_keyword() function pseudocode is as follows:

```
1. while true:
     a. n = get_new_random_id()
     b. digits = []
     c. while n:
          i. digits.append(int(n%62))
          ii. n //= 62
     d. digits = digits[::-1]
     e. for idx, val in enumerate(digits):
          i. digits[idx] =
             ALPHABET[val]
     f. keyword = "".join(digits)
     g. if keyword is unique: break
2. return keyword
```

The while-loop in the pseudocode above should take only one loop and a unique keyword is then returned.

2) User-chosen Keyword

The pseudocode for creating singkat object with user-chosen keyword is as follows:

```
1. If target_url is not valid: raise error 'Cannot
   access target URL'
2. If target_url is singkat: raise error 'Already a
   Singkat URL'
3. If keyword is not valid: raise error 'Keyword is
   not valid'
4. If keyword is not unique: raise error 'Keyword
   is already taken'
5. new_singkat = Singkat(target_url,
   keyword=remove_invalid_url_char(keyword)).
```

The remove_invalid_url_char() function will remove any invalid character in the keyword for it will be used as a URL segment. Total 71 characters are considered valid, based on RFC 1738 (without apostrophe and plus characters) [10], so there are $\sum_{n=1}^{100} 71^n$ keyword possibilities. By default, only registered users can create singkat object with chosen keyword. It is a way to minimize phishing and other misuses, such as discussed in [11].

D. Algorithm for Accessing Singkat Object

Singkat acts as intermediary to gather data when a singkat object is accessed by client (clicker), so the owner of that singkat object can monitor the click data, which includes ip address, geolocation, number of total and unique access, and timestamps. By default, appending '+' character to singkat URL will preview it, allowing the client to know where he/she will be redirected.

The algorithm is as follows:

```
1. If clicker is new:
     a. new_clicker = Clicker(ip,
        geolocation_data)
     b. new_click = Click(clicker, singkat)
     c. new_click_detail =
        ClickDetail(new_click, timestamp)
2. Else:
     a. If click is new:
          i. new_click = Click(clicker,
             singkat)
          ii. new_click_detail =
              ClickDetail(new_click,
              timestamp)
     b. Else:
          i. click = Click(clicker, singkat)
          ii. click.number_of_access += 1
          iii. new_click_detail =
               ClickDetail(click, timestamp)
```

The system will check whether the client is new by looking up the IP address in database. If it is not found, then a new clicker object is instantiated from Clicker class, which requires two main parameters: IP address and geolocation data. If the clicker already exists, then the system will simply update the statistics (total access by that particular client and timestamp). The simplicity of Singkat keyword, especially the pseudo random one, makes it prone to brute-force search [12], which may lead to security and privacy vulnerabilities.

TABLE I. DETAIL OF SINGKAT MODEL

Field	Type	Options
id	int	Primary key
keyword	char	unique, not null, length = 100
target	char (url)	length = 2000
title	char	blank, length = 100
owner	int	Foreign key, null
created_at	datetime	

TABLE II. DETAIL OF CLICKER MODEL

Field	Type	Options
id	int	Primary key
ip	char (ip address)	unique
city	char	blank, length = 100
country	char	blank, length = 100
continent	char	blank, length = 100
latitude	float	default = 0
longitude	float	default = 0
created_at	datetime	

TABLE III. DETAIL OF CLICK MODEL

Field	Type	Options
id	int	Primary key
ip	int	Foreign key
singkat	int	Foreign key
times	int	default = 1

TABLE IV. DETAIL OF CLICKDETAIL MODEL

Field	Type	Options
id	int	Primary key
click	int	Foreign key
time	datetime	

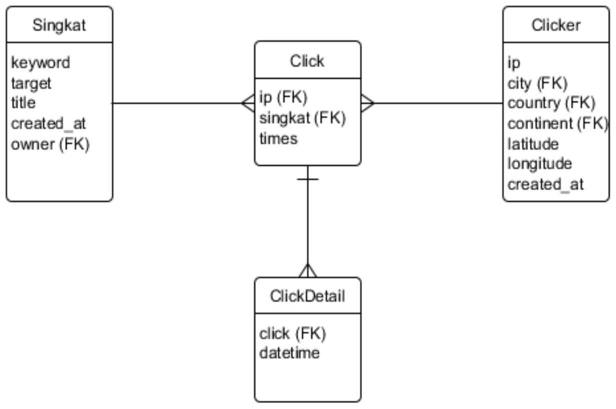


Fig. 1. Singkat model relationship diagram.

E. Implementation

The Django version used to build the package is 2.0.4 (released on April 2018). The package consists of one file containing view and helper functions, one file containing codes to define all models, one file containing URL patterns, several templates for web interface, additional helpers, and hooks. The total size of the package is 116.3 kB.

For showcase and testing purpose, we temporarily deployed the Singkat package in a Django web project located at http://praz.pythonanywhere.com. However, this hosting has whitelist policy so not every URL can be shortened. We chose to use SQLite as database management system and ipstack API as public IP-to-location service in the testing environment.

Fig. 2 and 3 show the web user interface, which allows users to interact with the package to do operations such as URL shortening, previewing Singkat URL, and accessing campaign statistics. It is currently styled using Bootstrap front-end framework for typography, forms, navigation, and other interface components.

A user who wants to create a new shortened URL will have to enter the original long URL and the keyword that he/she chooses in the web form, and then click 'Generate'. The user can also opt for automatically generated (pseudo random) keyword.

F. Testing

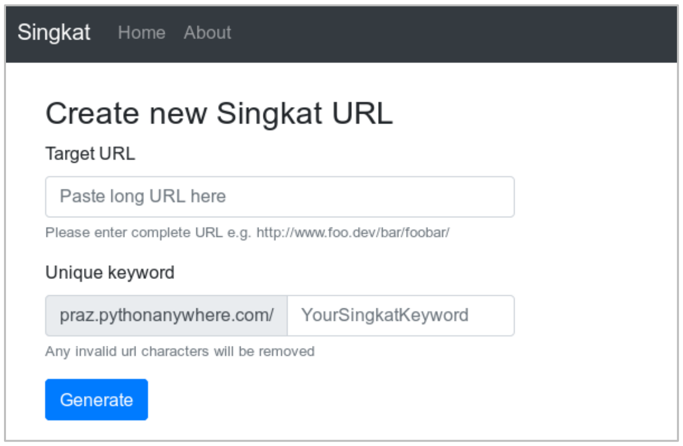
Three different tests are conducted to ensure the performance and functionality of Singkat:

1) Efficiency Testing

This test is to measure how much time it will take for Singkat algorithm to generate pseudo random keyword. As many users may try to shorten URLs at once, Singkat has to be able to generate keywords fast. Our target time is under 100 microseconds.

2) Behavioral Testing

This test is to make sure that there is no bug and unexpected behavior in Singkat by listing all possible behaviors and comparing them with actual results.



Create new Singkat URL

Target URL

Paste long URL here

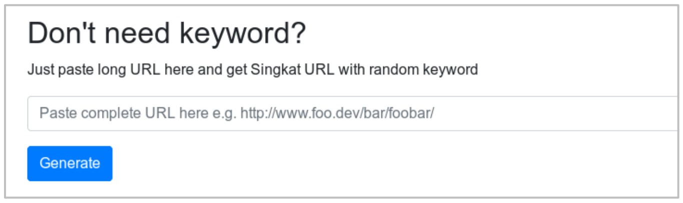
Please enter complete URL e.g. http://www.foo.dev/bar/foobar/

Unique keyword

praz.pythonanywhere.com/ YourSingkatKeyword

Any invalid url characters will be removed

Fig. 2. Singkat object creation form with user-chosen keyword.



Don't need keyword?

Just paste long URL here and get Singkat URL with random keyword

Paste complete URL here e.g. http://www.foo.dev/bar/foobar/

Fig. 3. Singkat object creation form with automatically generated keyword.

3) Online Campaign

This campaign is to mimic the real usage of Singkat URL shortener, by distributing a singkat object on various online media over a period, and then analyze whether all the click data have complete attributes, especially the geolocation and click count, so the owner of a singkat object may know whether his/her campaign has reached targeted people.

III. Results and Analysis

We measured how much time it took to generate a singkat object with pseudo random keyword using Python built-in datetime module. The long URLs as input were from various web page on the internet. We did not take into account the time it took for the 'requests' library to check the validity of long URL. We adjusted the value of current available pseudo random keyword identifier so the generated keyword from base62 conversion was reasonably short and contained multiple characters. We ran the test ten times, each to generate a new singkat objects. We found out that the average time needed to create ten new singkat objects was 0.0000436 second (43.6 microsecond). Because this test was conducted in a free hosting service, the performance should be better in fully dedicated server.

We ran a behavioral testing to ensure acceptable user experience and correct internal functionality based on the proposed algorithms. For example, Singkat package should not allow user to create a Singkat object that links to invalid URL. We excluded some non-core functions such as user registration and login. Table 5 shows expected and actual results. Valid means that the result is as expected, invalid otherwise.

As seen in Table 5, Singkat cannot identify singkat objects created by another Singkat package in different domain. This is a commonly accepted inconvenience, as many URL shortener services can still shorten already shortened URL from other service.

We also created and distributed a singkat object located at http://praz.pythonanywhere.com/TerminalFrost+ on various social media and online forums for research purpose. The experiment began on May 24, 2018 and ended on May 28, 2018. We then extracted all the click data from client area page (ten of them summarized in Table 6, with timestamp field omitted), and then we analyzed them. We managed to gather 156 unique click data; 30 (19.23%) of them had partially complete attributes which was commonly caused by clients accessing using cellular network, or could not be identified by public IP-to-location service. All of the data had complete click count and timestamp attributes, so time series graph could be made, which enabled the owner of the singkat object to track its popularity.

TABLE V. Behavioral Testing Result

Expected behavior	Actual result
User can create singkat object with pseudo random keyword	Valid
Registered user can create singkat object with user-chosen keyword	Valid
Registered user can access the detail of each of his/her singkat object	Valid
The system prevents creating singkat object that links to dead, unreachable, or invalid URL, and shows error notification to the user	Valid
The system shows error notification when a registered user tries to create a singkat object with already taken user-chosen keyword	Valid
The system automatically removes invalid URL characters in user-chosen keyword when creating singkat object	Valid
The system prevents creating singkat object that links to another singkat object, within the same domain	Valid
Same as above, but in different domain	Invalid
Client can access singkat object and be redirected to destination URL	Valid
Client can preview a singkat object before accessing it	Valid

TABLE VI. Sample Data from Research Campaign

IP Address	City	Country	Continent	Latitude	Longitude	Number of click
110.138.149.13	Bekasi	Indonesia	Asia	-62.349	1.069.896	1
61.247.32.164	Jakarta	Indonesia	Asia	-61.744	1.068.294	1
118.137.2.4	Bandung	Indonesia	Asia	-69.039	1.076.186	1
114.124.133.62	Jakarta	Indonesia	Asia	-61.744	1.068.294	1
52.42.129.76	Boardman	United States	North America	457.788	-119.529	1
52.40.20.219	Boardman	United States	North America	457.788	-119.529	1
114.124.234.63	Jakarta	Indonesia	Asia	-61.744	1.068.294	1
120.188.95.134	Jakarta	Indonesia	Asia	-61.744	1.068.294	1
114.124.179.38	Jakarta	Indonesia	Asia	-61.744	1.068.294	1
114.124.176.38	Jakarta	Indonesia	Asia	-61.744	1.068.294	1

IV. Conclusion

Performance and all core functions of Singkat had been tested using various method, with satisfying result. Power users can deploy Singkat package by importing it and setting up some configurations, e.g., it is up to the system administrator to choose and set up the database management system powerful enough to prevent locking, which could be caused by the increasing value of pseudo random keyword identifier and concurrent data access. In Django, this can be configured easily.

The Singkat package can be obtained at github.com/GottfriedCP/Singkat-URL-Shortener. It is licensed under free MIT License.

In the future, RESTful API feature should be developed to make the package more customizable and extensible. Integrated geolocation module should be developed, instead of relying on external IP-to-location service. Algorithms to create both user chosen and pseudo random keyword could also be improved to handle heavy, concurrent usage, and ensuring every keyword will always be unique.

Acknowledgment

We would like to thank Gunadarma Foundation for their support.

References

[1] Connolly D, Sperberg-McQueen CM. Web addresses in HTML 5. May 2009. [Online]. Available: https://www.w3.org/html/wg/href/draft#url. [Accessed 1 May 2018].

[2] Azar E, Alebicto ME. Swift Data Structure and Algorithms. Birmingham:Packt Publishing. 2016: 236.

[3] prolific. We want 'em shorter. July 2001. [Online]. Available: https://www.metafilter.com/8916/We-want-em-shorter. [Accessed 1 May 2018].

[4] Stack Overflow. Developer Survey Results 2018. [Online]. Available: https://insights.stackoverflow.com/survey/2018/#technology. [Accessed 1 May 2018].

[5] Robinson D. The Incredible Growth of Python. September 2017. [Online]. Available: https://stackoverflow.blog/2017/09/06/incredible-growth-python/?_ga=2.28880508.1369638144.1525141647-1574360211.1503055200. [Accessed 1 May 2018].

[6] George N. Mastering Django Core. Birmingham: Packt Publishing. 2016.

[7] Reitz K, Schlusser T. The Hitchhiker's Guide to Python: Best Practices for Development. Sebastopol:O'Reilly Media. 2016.

[8] Koehler W. A longitudinal study of Web pages continued: a report after six years. *Information Research*. 2004; 9(2): paper 174.

[9] Fielding R, Reschke J. RFC 7230 - Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing. June 2014. [Online]. Available: https://tools.ietf.org/html/rfc7230#section-3.1.1. [Accessed 2 May 2018].

[10] Berners-Lee T, Masinter L, McCahill M. RFC 1738 - Uniform Resource Locators (URL). December 1994. [Online]. Available: https://www.ietf.org/rfc/rfc1738.txt. [Accessed 2 May 2018].

[11] Klien F, Strohmaier M. *Short links under attack: geographical analysis of spam in a URL shortener network*. Proceedings of the 23rd ACM conference on Hypertext and social media. 2012; 83-88.

[12] Georgiev M, Shmatikov V. Gone in six characters: Short urls considered harmful for cloud services. *arXiv*. 2016; arXiv:1604.02734.