Contents:
CHAPTER 1

Installation

1.1 Requirements

- golang
- python-requests module (to add sample components to play with)
- mariadb server
- redis
- git (to get the sources)

1.2 Install golang

Download golang from here, extract go directory under your home directory.

$ mkdir ~/gocode

Now write the following lines in your ~/.bashrc file.

export PATH=$PATH:~/go/bin
export GOPATH=~/gocode/
export GOROOT=~/go/

and then

$ source ~/.bashrc

1.3 Install the dependencies

After golang installation, get the dependent libraries.

$ go get github.com/garyburd/redigo/redis
$ go get github.com/go-sql-driver/mysql
$ go get github.com/vaughan0/go-ini
1.4 Setup Mariadb (or MySQL)

$ mysql -u root
> CREATE USER 'bugspad'@'localhost' IDENTIFIED BY 'mypass';
> CREATE DATABASE bugzilla;
> GRANT ALL PRIVILEGES ON bugzilla.* TO 'bugspad'@'localhost';

1.5 Clone the git repo

Now clone the source repo somewhere in your home directory.

$ git clone https://github.com/kushaldas/bugspad.git

1.6 Create the tables

First edit scripts/bootstrap.sql line 2 with your username and email id.

$ mysql -u bugspad -pmypass bugzilla < createdb.sql
$ mysql -u bugspad -pmypass bugzilla < bootstrap.sql

1.7 Build bugspad

$ make

After this you have to build the helper tools also.

$ go build load_all_bugs_redis.go redis_op.go backend.go

This should create a binary called bugspad in the directory.

1.8 Install and run redis server

# yum install redis
# service redis start

1.9 Customize config file

First, copy the sample config file config/bugspad.ini-dist to config/bugspad.ini.

$ cp config/bugspad.ini-dist config/bugspad.ini

Now, edit config/bugspad.ini and add proper credentials(user and password) to access your bugzilla database.
1.10 Start the backend server

First run the loader to load all index data in redis.

$ ./load_all_bugs_redis

$ ./bugspad

1.11 Populate database with components

So, we will put some (16k+) components in the database so that we can test.

$ cd scripts
$ wget http://kushal.fedorapeople.org/comps.json.tar.gz
$ tar -xzvf comps.json.tar.gz

Then update addcomponents.py with your email id as username and execute it.

$ python addcomponents.py
We are using redis to store all search related indexes on memory. This means any search term must be indexed in redis.

2.1 File structures

- **bugspad.go** contains all web code
- **backend.go** contains all logic code
- **redis_op.go** contains all redis operation functions
- **load_all_bugs_redis.go** contains helper code to create index on redis
The following document explains the current Web API, remember this project is under heavy development, so the API inputs might change a lot.

### 3.1 Creating a new component

- **Request type:** POST
- **URL:** /component/

Post data:

```json
{
    "description": "description of the component",
    "name": "Name",
    "product_id": 1,
    "user": "user@example.com",
    "password": "asdf",
    "owner_id": 1
}
```

### 3.2 Get component list for a product

- **Request type:** GET
- **URL:** /components/<int: product_id>

Output:

```json
{
    "0ad": [
        "522",
        "0ad",
        "Cross-Platform RTS Game of Ancient Warfare"
    ],
    "0ad-data": [
        "523",
        "0ad-data",
        "The Data Files for 0 AD"
    ],
    "0xFFFF": [null]
}
```
3.3 Create a new bug

- Request type: POST
- URL: /bug/

Post data:

```json
{
  "user":"username@example.com",
  "password":"asdf",
  "summary":"summary text of the bug",
  "description":"description of the bug",
  "component_id":1,
  "subcomponent_id":1,
  "status":"status of the bug",
  "version":"version",
  "severity":"severity",
  "hardware":"hardware",
  "priority":"priority",
  "whiteboard":"whiteboard",
  "fixedinver":null
}
```

Output:

`bug_id`

3.3.1 Default values (optional arguments)

`priority`, `severity` has a default value of “medium”. `status` is “new” by default. `hardware`, `whiteboard`, `fixedinver`, `subcomponent_id` is optional.

3.4 Update a bug

- Request type: POST
- URL: /updatebug/

Post data:

```json
```
3.5 Adding a comment to a bug

- Request type: POST
- URL: /comment/

Post data:

```json
{
    "user": "username@example.com",
    "password": "asdf",
    "bug_id": 1,
    "desc": "comment text",
}
```

3.6 Getting details of a bug

- Request type: GET
- URL: /bug/<int bug_id>
Indices and tables

- genindex
- modindex
- search