SohoMuse

Overview

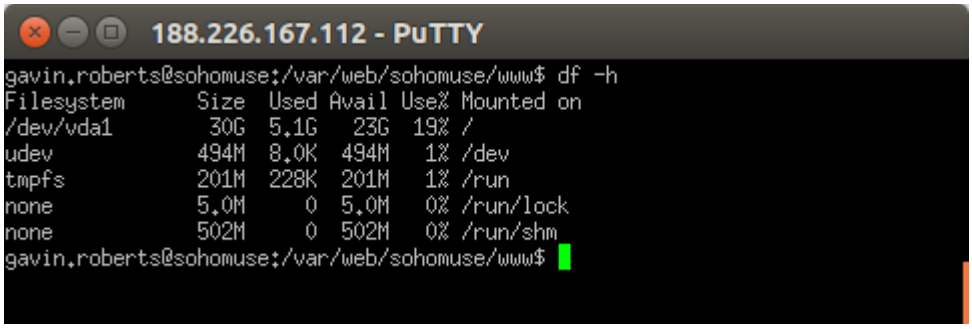
This document hopes to provide as much information as possible to the reader to understand the functionality available on the SohoMuse website. This document is to compliment the documentation already in place on the BitBucket repository.

Server

The SohoMuse website is currently being hosted by a Virtual Private Server within the Digital Ocean network.

IP: 188.226.167.112
OS: Ubuntu 12.04.4 (32Bit)
RAM: 1GB
HDD: 30GB SSD
Region: Amsterdam 2
Backups: Weekly Wednesday 7pm UTC.

Current disk usage:



The above includes the operating system.

Architecture

The SohoMuse website consists of two area's.

Frontend

Name	Version	Description
jQuery	~1.11.1	Base javascript framework.
Backbone	~1.1.0	Data binding framework.
Underscore	~1.6.0	Helper library.
Normalize-css	~3.0.1	Reset's all browsers to appear the same.
RequireJS	~2.1.14	Lazy loading framework.
RequireJS-text	~2.0.12	RequireJS add-on to load text files as templates.
Bootstrap	3.1.0	Grid/Frontend framework.
Require-css	~0.1.4	RequireJS add-on to load and compile CSS.
Marionette	~1.8.6	Javascript MVC framework.
Selectize	~0.9.1	jQuery tagging plugin.
Blueimp-file-upload	~9.5.7	jQuery file upload plugin.
Backbone-query-parameters	~0.3.0	Backbone add-on to support querystring parameters.
Backbone-poller	~0.2.8	Backbone add-on to support polling of a datasource.
Moment	~2.6.0	Javascript date parser and formatter..
Backbone.eventbinder	~1.0.2	Backbone add-on to support event binding.
JQuery-ui	~1.10.3	Base UI Javascript framework.
Backbone-relational	~0.8.8	Backbone add-on to support relational data.
Imgareaselect	~0.9.11	jQuery image crop plugin.
jQuery.cookie	~1.4.1	jQuery cookie plugin.

The frontend consists of an MVC Single page application which communicates using Ajax with the API backend.

All styles are based on the Bootstrap UI Framework which provides a mobile frist grid system and are then extended on a per page basis using the Less CSS compiler.

Backend (API):

Name	Version	Description
Agenda	^0.5.10	NodeJS Scheduler.
Async	~0.7.0	Asyncronous functionality.
Connect-redis	~1.4.5	Redis session storage.
Csso	^1.3.11	CSS Optimiser.
Ejs	~0.8.4	Embedded javascript templating system.
Email-templates	~0.1.0	Email templating system.
Express	~3.3.8	NodeJS Framework.
Fs.extra	~1.2.1	Enhanced filesystem support.
Gm	~1.12.2	Image processing for NodeJS.
Mmmagic	~0.3.4	Content type detection.
Mongoose	~3.6.20	MongoDB data connector.
Mongoose-fts	~0.2.0	Full text search support for MongoDB.
Node-foursquare-venues	0.0.6	Foursquare API wrapper.
Nodemailer	~0.5.2	Framework to send emails.
Passport	~0.1.17	NodeJS Authentication framework.
Passport-http-bearer	~1.0.1	HTTP Bearer authentication add-on for Passport.
Passport-local	~0.1.6	Simple authentication add-on for Passport.
Passport-local-mongoose	~0.2.5	MongoDB authentication add-on for Passport.
Redis	~0.8.6	Redis data connector.
Requirejs	~2.1.8	Lazy loading framework.
Restler	~2.0.1	HTTP client for NodeJS.
Underscore	~1.5.2	Helper library.
Webshot	~0.14.0	Screenshot library.
Xml2js	~0.2.8	Convert XML to Javascript library.
Date-format-lite	~0.5.0	Date parsing and formatting library.

The backend acts as an restful-like API in which the frontend subscribes to. The idea behind the API is to easily allow for the replacement of and addition of frontend's to the data.

The API is primarily restricted to authenticated users, using the Passport authentication NPM package. Should a request be made to a authenticated route without a valid user, a 401 http header is returned.

Features

By combining the front and backend together via the API, the SohoMuse website provides the following features:

- Authentication
  - Closed registration.
  - Recover forgotten username/password.
- Social Networking
  - Connections
  - Newsfeed
  - Like/Commenting
  - Endorsements
  - Messaging
  - Sharing
  - Invitations
- Professional
  - Availability
  - Representation
  - Social
- Media
  - Images
  - Videos
  - Files
  - Showreel
  - Business card

Underlying functionality also includes the ability to find connections close to you based on your reported geo-location (browser based location.)

Installation

Simply connect to the server you wish to install SohoMuse on and run the following:

```
1. wget  https://[username]:
   [password]@bitbucket.org/NavadaDev/soho/raw/master/docs/install.sh
2. sudo chmod +x install.sh
3. sudo ./install.sh [path to install to] [username] [password]
```

Replace the following variables with their respective values:

[username] – BitBucket username
[password] – BitBucket password
[path to install to] – The path in which you would like to install SohoMuse.

e.g.

```
wget https://username:password@bitbucket.org/NavadaDev/soho/raw/master/docs/install.sh
sudo chmod +x install.sh
sudo ./install.sh ~/sohomuse/www username password
```

MongoDB

Although it's possible to use an GUI interface for MongoDB, using the command line can sometimes be a lot quicker.

Connecting to the database:

```
mongo // load the mongodb client
use soho // switch to the soho database.
show collections // display the list of tables
db.users.find({ username: 'umi' }); // only show results where username = Umi
```

Exporting data:

```
mongoexport --db soho -out backup.json
```

Import data:

```
mongoimport --db soho -file backup.json
```

Please refer to the user manual for more information: http://docs.mongodb.org/manual/

Sitemap

http://www.gliffy.com/go/publish/8212815

Analytics

Url: https://analytics.google.co.uk
Username: sohomuse2015
Password: S0h0Mu53

Due to how SohoMuse is architectured, as you perform actions throughout the site, it does not initiate a “page load” event which Google Analytics uses to identify a user and their activity.

As a result of this, we are only able to log connections to the site based on the API usage, i.e. how many people are using the API that feeds the front end of the website.

Due to the number of different interaction points throughout the application, it would take a considerable amount of planning and integration to add events showing interactions such as page views, interactions (i.e. x messaged y, y connected with x), events (i.e. error sending message) etc.

Deploying changes

As node servers do not automatically restart should an error occurr, Forever is used to monitor a node process and if it exits, it will automatically attempt to restart it.

Node keeps a copy of the code it's executing in memory, which means you will be required to restart the node process after pulling the latest changes.

```
1. Cd [path to SohoMuse root]
2. git pull
3. forever restartall
```

Initiate the website

Should the server it self reboot, either due to maintenance or to an unknown error, the website will not automatically start serving like a traditional Apache based server would.

As a result, you will need to initiate the website:

```
1. Cd [path to SohoMuse root]
2. forever server.js logs/node.log &
```

Note: running #3 of the above will result in all of the output from the server (i.e. logs etc) to be shown on the screen. See the `screen` section below for how to best deal with this.

Using Screen

Screen is used on the SohoMuse to manage multiple terminal sessions from the SSH session.

```
1. screen -r
2. CTRL+A + [number of screen]
   0. Log
   1. Git
   2. Forever
3. CTRL+D // detach and return to SSH session.
```

Using screen allows you to initiate the website in the Log window (CTRL+A + 0) and view the log and then easily switch to the Git window (CTRL+A + 1) to perform any Git/File system functions on the website source.

Use of Screen is not absolutely required, however it is advised.