TEAM MEMBERS

Alon Sigal 998156846 g1sigal
Henry Ku 998551348 g2kuhenr
Simon Joon-Hee Song 998447006 g2junhee
Zheng (Lionheart) Xiong 998182112 c3xiongz

CDF ENVIRONMENT

greywolf.cdf.toronto.edu:31365

ADDITIONAL MODULES INSTALLED

By using npm install:

- Request
- Mysql
- cron
- querystring

CODE DESIGN

Method: **POST /blog**
Parameters:
- blog: a string indicating a new blog to track by its {base-hostname}

Response: HTTP status 200 if accepted

First we parse the parameter 'blog' to find out which bloggers to track. It is done through line 560 – line 565. After that first we check if this blogger is already being tracked by checking its existence in BLOG database table. If it does exist, we do not have to do anything else. If it does not exist, we add 'blog' to our BLOG database table and call insertLikes(blog).

In insertLikes function, we use API call from tumblr to retrieve all post liked by the 'blog'. Request module in Node.JS is used to retrieve liked posts by calling request.get with **http://api.tumblr.com/v2/blog/'+hostname+'/likes?api_key='+KEY+'&limit=50** url. Hostname is an variable that holds 'blog' and KEY is API key we received from tumblr. Then insertLikes retrieves 'liked_count' data from API and send it to insertLikeHelper function along with hostname.

insertLikesHelper calls tumblr API using request.get, just like insertLikes function did, with **http://api.tumblr.com/v2/blog/'+hostname+'/likes?api_key='+KEY+'&limit=50&offset='+off** We set limit to 50, because it is the maximum number of posts that this API can bring in. and we repeat this process x amount time (x = liked_count), and save every liked posts into our data table. This is done by increasing offset value by 50 every loop. (Offset specifies which post we start to count 50 from)

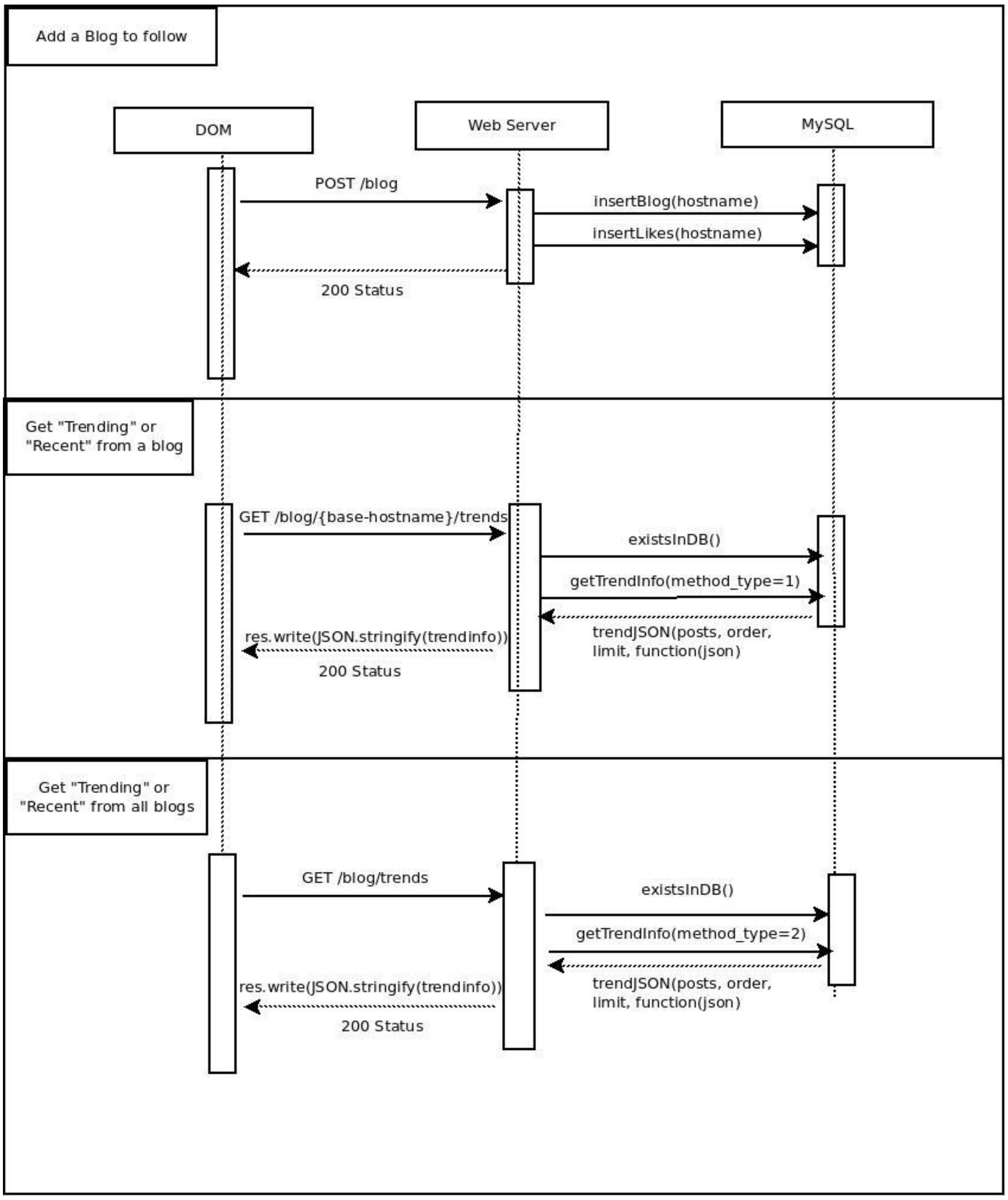
From the structure that tumblr API returned, {post_url} is saved for url, where the posts is located, {data} is saved for date, which indicates when the post was uploaded, {slug} is saved for description, basically it is an title of a post, {image-permalink} is saved for image if exists, and lastly {note_count} is saved for note_count which specifies how many likes this post have.

Method: **GET /blog/{base-hostname}/trends**
Method: **GET /blogs/trends**
Parameters:
- limit: the maximum number of results to return (optional)
- order: a string "Trending" or "Recent" to determine which posts should be included in the response. "Trending" should return those posts that have the largest increments in note_count in the last hour, while "Recent" should return the most recent posts regardless of their popularity.

To retrieve "Recent" data, we simply manipulate our datatables by 'dt' value (specifies the date of the post is posted) and retrieve 'limit' number of posts. The default limit is 20. If specific {base-hostname} is provided, we ignores everything else but the posts liked by {base-hostname}.

To retrieve "Trending" data, we organize the order by greatest number of 'inc' value in time_stamp table. First we joined time_stamp table and post table, and find greatest 'seq' value of each posts and compare their 'inc' value to retrieve posts who has greatest increase in note_count within an hour. If {base-hostname} is provided, we only consider posts liked by {base-hostname}.

SEQUENCE DIAGRAM


Add a Blog to follow
DOM
Web Server
MySQL
POST /blog
insertBlog(hostname)
insertLikes(hostname)
200 Status
Get "Trending" or "Recent" from a blog
GET /blog/{base-hostname}/trends
existsInDB()
getTrendInfo(method_type=1)
trendJSON(posts, order, limit, function(json)
res.write(JSON.stringify(trendinfo))
200 Status
Get "Trending" or "Recent" from all blogs
GET /blog/trends
existsInDB()
getTrendInfo(method_type=2)
trendJSON(posts, order, limit, function(json)
res.write(JSON.stringify(trendinfo))
200 Status

Database Diagram

BLOG

column	property	type	size limit	reference from
url	primary	char	500	/

POST

column	property	type	size limit	reference from
url	primary	char	500	/
blog_url	not null	char	500	/
txt	/	char	500	/
img	/	char	500	/
dt	not null	timestamp	/	/

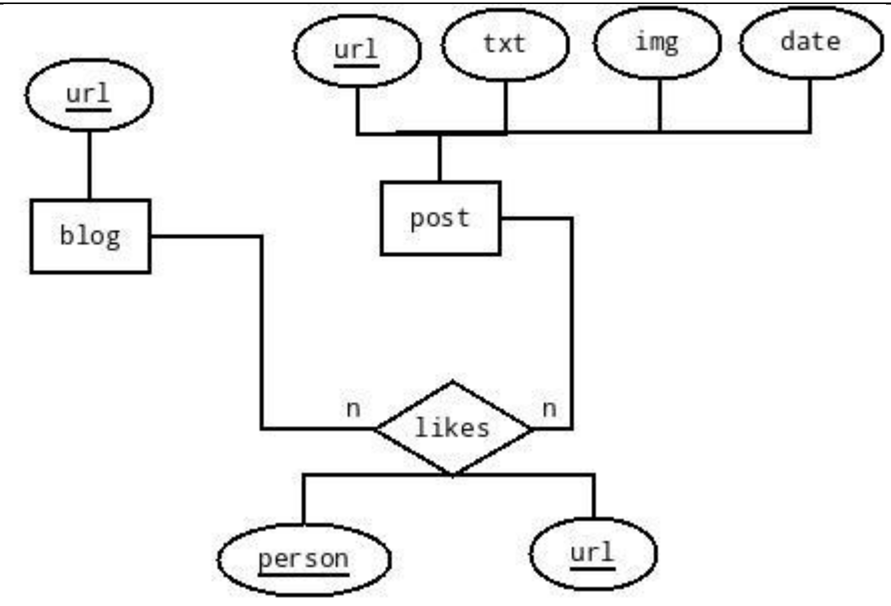
TIME_STAMP

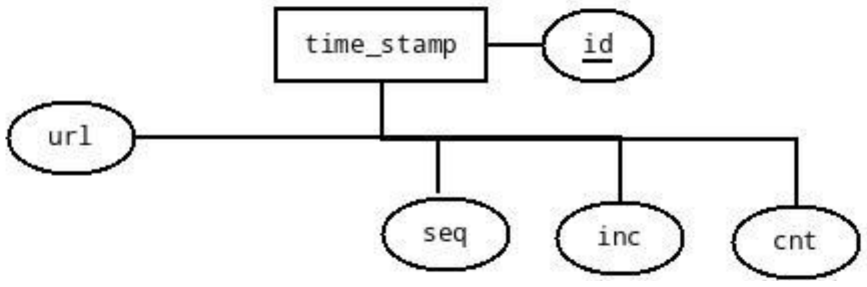
column	property	type	size limit	reference from
id	primary	integer	/	/
ts	/	timestamp	/	/
url	/	char	500	POST 'url'
seq	/	integer	/	/
inc	/	integer	/	/
cnt	/	integer	/	/

LIKES

column	property	type	size limit	reference from
url	foreign key, not null	char	500	POST 'url'
person	foreign key, not null	char	500	BLOG 'url'

ERD


url
txt
img
date
url
blog
post
n
likes
n
person
url


time_stamp
id
url
seq
inc
cnt