Rotzy

Building an iPhone Photo-Sharing App on Google App Engine
www.rotzy.com

by Gee-Hwan Chuang

What is Rotzy?

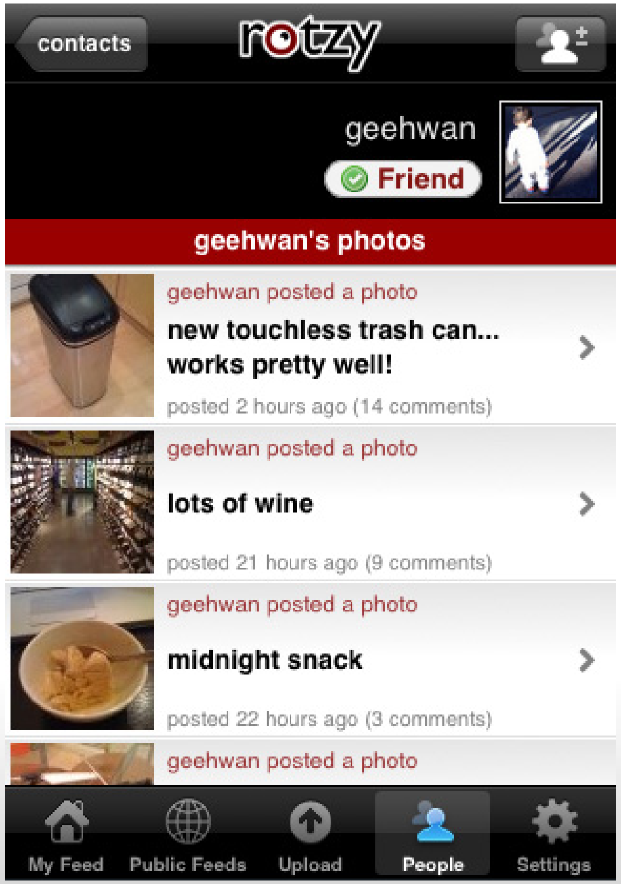
- Rotzy is a mobile community for instantly sharing, discussing and discovering photos around the world with location tagging and maps.
- After you take and upload a photo from your device, it becomes part of your gallery and is broadcast to your friends and followers in a Twitter-like timeline.
- Users can discuss your photo and even reply with another photo for a richly interactive experience.
- You can also simultaneously broadcast photos to other services like Twitter, Facebook, Tumblr, or your own web site using our widget.

Interesting Hurdles

- Photo Storage
- Location Search
- Random Byte Access (for Videos)
- Background Tasks

Photo Storage

- Photos are currently small and are stored as blobs.
- Thought about moving them to S3 but not worth it yet.
- It costs more to store it in the datastore because of datastore overhead as well as cpu usage while retrieving photos.
- One benefit is that it gives us a trigger for doing background tasks, since there is currently no way to do this natively.



geehwan posted a photo
new touchless trash can... works pretty well!
posted 2 hours ago (14 comments)

geehwan posted a photo
lots of wine
posted 21 hours ago (9 comments)

geehwan posted a photo
midnight snack
posted 22 hours ago (3 comments)

geehwan posted a photo

Photo Storage

Sample Model for Storing Photos

```
class PhotoData(db.Model):
    photo_o = db.BlobProperty()
    photo_l = db.BlobProperty()
    photo_m = db.BlobProperty()
    photo_s = db.BlobProperty()
    photo_t = db.BlobProperty()
    created = db.DateTimeProperty(auto_now_add=True)
```

Location Search

Some relational databases have built-in methods or extensions that allow for easy latitude, longitude searches.

The AppEngine datastore does not... and on top of that you can only do an inequality filter on one property.

That means, you cannot do something like,

```
WHERE lat > x1 AND lat < x2 AND lon > y1 AND lon < y2
```

Location Search

So, we use a python implementation of Geohash
http://en.wikipedia.org/wiki/Geohash

Essentially, this creates a string hash based on a given latitude and longitude.

A property of this hash is that places that are close together have a similar prefix. The longer the prefix, the closer they are.

This hash is calculated and stored along with the latitude, longitude data.

http://mappinghacks.com/code/geohash.py.txt

Location Search

An Example,

Cupertino, CA : **9q9h**qc4k35xxn

Mountain View, CA: **9q9h**y0t1kzgxg

Los Angeles, CA: **9q**5ctr1b27xzt

Austin, TX: **9**v6kpy205t1t6

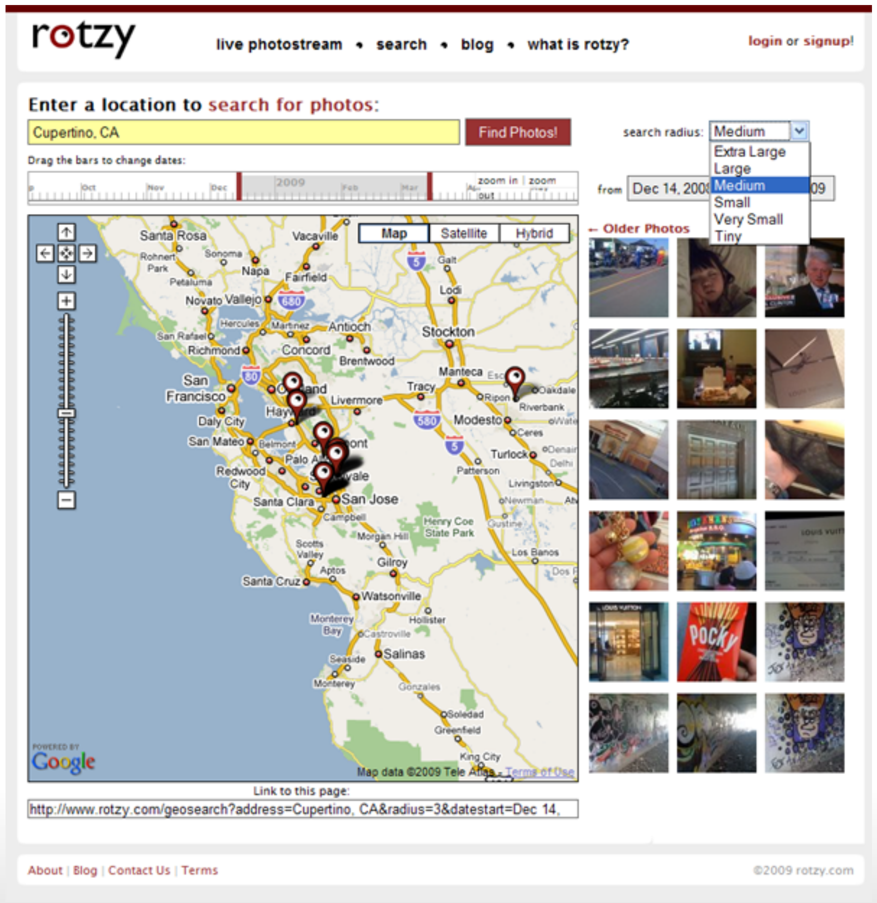
Boston, MA: drt2zp1q95084

Location Search

How the query works...

Given the search "radius",
we do a prefix match of up to
6 characters. radius=1,2,3, ...

For speed, we store each of these
6 possible prefixes in the db.
bbox1, bbox2, bbox3, ...



Enter a location to search for photos: Cupertino, CA

search radius: Medium

from Dec 14, 2008

Link to this page:
http://www.rotzy.com/geosearch?address=Cupertino, CA&radius=3&datestart=Dec 14,

```
hash = str(geohash.Geohash((lon, lat)))
bboxhash = hash[:radius]
photos = Photos.all().filter('bbox'+str(radius)+' = ',bboxhash)
photos = photos.filter('created >= ',startDT).filter('created <= ',endDT).order('-created').fetch(20)
```

Random Byte Access

We are experimenting with storing short video clips in the datastore along with photos.

When serving an mp4 video clip, the iPhone will use HTTP random byte access to get parts of the file.

Many regular web servers handle this seamlessly for files.

Since the video clip is stored as a blob... we need to check the HTTP headers to see which piece of the data needs to be returned and only return that portion of the blob.

Random Byte Access

```
headers = self.request.headers
if headers.has_key('Range'):
    start_byte,end_byte = (headers['Range'].split('='))[1].split('-')
    start_byte = int(start_byte)
    if end_byte == "":
        end_byte = 0
    else:
        end_byte = int(end_byte)
    ranged = True
else:
    ranged = False

# Set Response Headers
self.response.headers['Content-Type'] = "video/mp4"
self.response.headers['Accept-Ranges'] = "bytes"
if ranged:
    self.response.set_status(206)
    content_len = str(len(content))
    if end_byte == 0:
        self.response.headers['Content-Range'] = "bytes " + str(start_byte) + "-" + content_len + "/" + content_len
    else:
        self.response.headers['Content-Range'] = "bytes " + str(start_byte) + "-" + str(end_byte) + "/" + content_len

# Write out content
if ranged:
    if end_byte == 0:
        self.response.out.write(content[start_byte:])
    else:
        self.response.out.write(content[start_byte:(end_byte+1)])
else:
    self.response.out.write(content)
```

Background Tasks

You cannot run background tasks on AppEngine yet.
You can only run code in response to queries... so we just do a little bit on queries that otherwise do not take very long.

For Rotzy, that means photo requests and profile pic requests.

We created a "TaskQueue" to store the pending tasks and do one single task each time a profile photo is requested.

Background Tasks

```
class TaskQueue(db.Model):
    task_type = db.StringProperty(required=True)
    param1 = db.StringProperty(required=True)
    param2 = db.StringProperty(required=True)
    created = db.DateTimeProperty(auto_now_add=True)

    # "atomically" get the next task
    def getNextTask(key):
        task = db.get(key)
        if task:
            ret = {"task_type":task.task_type, "param1":task.param1, "param2":task.param2}
            task.delete()
            return ret
        else:
            return None
    getNextTask = staticmethod(getNextTask)

    def doTask():
        # the next task to do as a transaction
        task = TaskQueue.all().order('created').get()
        if task:
            try:
                tasktuple = db.run_in_transaction(TaskQueue.getNextTask, task.key())
            except:
                tasktuple = None
            if tasktuple:
                 if tasktuple["task_type"] == "mktwit":
                    TaskQueue.mkTwit(tasktuple["param1"], tasktuple["param2"])
    ...
```

Thank You and Contact Info

Check out Rotzy on the web or iPhone:

http://www.rotzy.com

Some other projects I'm working on also hosted in the cloud:

http://www.remobo.com
http://www.solecial.com

My contact info if you have questions about Rotzy, Google App Engine, or anything else:

Gee-Hwan Chuang, http://www.geesblog.com, follow me on twitter @geehwan