Piperpal

Piperpal and the <location> Content Tag

Version 2

Ole Aamot

ole@aamotsoftware.com

https://piperpal.com/

Background

I’m a Norwegian native currently staying in Palo Alto.

I have worked on piperpal.com for the last 2 weeks and the concept of the <location> Content Tag for 4 years.

I quit my day job to implement Piperpal in August 2015 and I went to Palo Alto, California to meet with Peter Norvig, Director of Research, Google, Inc. to discuss a new possible <location> Content Tag on piperpal.com.

Norvig is very positive to the concept of crowd-sourced location data entry with a location tag for content and the idea of building my new website such as Piperpal as a prototype for location-based search using the formula known as Haversine.

He said that Google may implement it too, but it would be based on the Geo strategy at Google and implemented in the terms of Google’s infrastructure.

In 3 years I have built piperpal.com, a web site where you can add content based on the geographical position of your web browser and pay using the stripe.com APIs.

On piperpal.com you can search for data based on text queries, with autocomplete of existing entries, in a radius in the range of 0 - 10000 km from your current location.

A <location> tag for location-based markup

I am introducing the new <location> tag and &<location> syntax for document markup of location-based content in HTML by creating the Data Type Definition published on http://github.com/location

The new tag will make it easier for content providers to state that the HTML content they are publishing is geographically positioned and meant for indexing and displaying on services like Google Maps / Google Now.

The motivation behind this work is new location-based content retrieval, and would open a range of new ways to advertise geographically within a given radius and time interval.

Example: &concert

Consider a classical concert at The Greek Theater. The location tag is &concert.

```
<!DOCTYPE location SYSTEM
"https://raw.githubusercontent.com/location/location/m
aster/location-1.3.dtd">

<location name="concert" data="The Greek Theater"
link="http://www.ticketmaster.com/" glat="37.873596"
glon="-122.25443" radius="10000"
notBefore="2017-12-31T22:00"
notAfter="2018-01-01T00:00" paid="50"/>
```

JavaScript API for Location on piperpal.com

I built an API for location-based index on piperpal.com.

Example Site: https://piperpal.com/paloalto.html

```
<script type="text/javascript"
src="https://api.piperpal.com/location/json.php?service
=Search&glat=37.4375596&glon=-122.11922789999998"></scr
ipt>
<script language="JavaScript">
var obj = JSON.parse(locations);
document.write(obj.locations[0].distance + " " +
obj.locations[0].name + " " + obj.locations[0].location
+ " " + obj.locations[0].service + " " + "<br />\n");
document.write(obj.locations[1].distance + " " +
obj.locations[1].name + " " + obj.locations[1].location
+ " " + obj.locations[1].service + " "+ "<br />\n");
</script>
```

Result of Location API Search:

Location Tags in Palo Alto, CA

3.2512275632216996 GoogleVisitorCenter http://www.google.com/ Search

3.259510067275075 Google Visitor Center http://www.google.com/ Search

Piperpal: Location-aware Content markup

My idea is that the content on a website is marked up according to a location tag and that this tag decides which content that is meant to be indexed and eligble for placing a ad on. The content producers would tag their content with location tags, advertisers mark up their catalog with location-aware radius ads, and the users can embed the ads in their posts with a new location tag syntax such as &concert that could be implemented for Google services such as Gmail and Google+.

Pseudo code for logic of search for nearby matches

```
if (notBefore < NOW() < notAfter || ((&UserLoc - radius) < (geo) < (&UserLoc + radius)) → display
```

Piperpal is the first site that I am implementing for making a resource for location-based tags.

On https://piperpal.com/ I added a form that lets the user insert the Name, Location, and Service parameter and pay by credit card (via Stripe) to add the entry.

The mapping between the content produced by a content provider, the ad by the advertiser and the tagging by individual user is done on www.piperpal.com/<location> where the location tag example for <location> would be &concert and the URI would be www.piperpal.com/concert

Actual implementation

```
CREATE TABLE piperpal (
       id MEDIUMINT(8) UNSIGNED NOT NULL AUTO_INCREMENT,
       name VARCHAR(100) NOT NULL DEFAULT '',
       service VARCHAR(1024) NOT NULL,
       location VARCHAR(1024) NOT NULL,
       modified TIMESTAMP NOT NULL DEFAULT
       CURRENT_TIMESTAMP ON UPDATE CURRENT_TIMESTAMP,
       created TIMESTAMP NULL DEFAULT NULL,
       glat DECIMAL(10, 8) NOT NULL,
       glon DECIMAL(11, 8) NOT NULL,
       paid MEDIUMINT(16) UNSIGNED NOT NULL,
       token VARCHAR(32) NOT NULL,
       type VARCHAR(32) NOT NULL,
       email VARCHAR(256) NOT NULL,
       PRIMARY KEY (id));
);
```

Apache mapping: https://piperpal.com/concert

```
RewriteEngine On
RewriteRule ^([\sa-åA-Å0-9]+)$ /api/ [L]
```

jQuery representation of "https://piperpal.com/concert"

Piperpal provides a jQuery function for variables in a the list of all location entries for a given location tag nearby a geographical position for the device.

This jQuery function can then be used by third-party developers who want to show all entries for a given location-based tag entry accepted on piperpal.com.

Piperpal Location jQuery API

The API will query the user’s link location and geoposition through the AJAX script and present a user with content in the nearest geographical approximity according to the Haversine formula.

```
<div id="log"></div>
<script>
   $(document).ready(function(){
        setInterval(function(){
          if (navigator.geolocation) {
              navigator.geolocation.getCurrentPosition(ajaxCall);
          } else{
              $('#log').html("GPS is not available");
          }
          function ajaxCall(position){
            var latitude = position.coords.latitude;
            var longitude = position.coords.longitude;
            var location = window.location.pathname.substr(1);
            $.ajax({
                     url: "/api/pull.php",
                     type: 'POST', //I want a type as POST
                     data: {'latitude': latitude, 'longitude' : longitude,
                            'location' : location },
                     success: function(response) {
                        $('#log').html(response);
                     }
              });
            }
          },1500);
         });
</script>
```

Formula for computing a Haversine distance

The distance along the surface of the (spherical) Earth between two arbitrary points, in degrees, is determined by the Spherical Cosine Law, also known as the Haversine Formula. We use this law to compute the nearest entries on piperpal.com measured from the user’s geolocation.

```
SELECT DISTINCT
id,name,service,location,modified,created,glat,glon,pa
id,token,type,email,111.045*DEGREES(ACOS(COS(RADIANS(l
atpoint))*COS(RADIANS(glat))*COS(RADIANS(longpoint)-RA
DIANS(glon))+SIN(RADIANS(latpoint))*SIN(RADIANS(glat
)))) AS distance_in_km FROM piperpal JOIN (SELECT  " .
$_POST['latitude'] . " AS latpoint, " .
$_POST['longitude'] . " AS longpoint) AS p ON 1=1
WHERE name = '" . $_POST['location'] . "' ORDER BY
distance_in_km;
```

The Piperpal Location JSON function is on
https://api.piperpal.com/location/json.php?service=Search&glat=37.44&glon=-122.12

```
var locations = '{ "locations" : [' + '{"id": "2", "name":
"GoogleVisitorCenter", "service": "Search", "location":
"http://www.google.com/", "modified": "2018-03-16 02:23:31", "created":
"2015-08-27 16:29:49", "glat": "37.42281050", "glon": "-122.08737760", "paid":
"1", "token": "tok_16eTG3AZBHUS3EAZUcuZKov5", "type": "card", "distance":
"3.452303131168993", "email": "oka@oka.no"},{"id": "1", "name": "Google Visitor
Center", "service": "Search", "location": "http://www.google.com/", "modified":
"2018-03-16 02:23:28", "created": "2015-08-27 16:27:16", "glat": "37.42242580",
"glon": "-122.08755550", "paid": "1", "token": "tok_16eTDaAZBHUS3EAZZJ4MCZFK",
"type": "card", "distance": "3.463140792214949", "email": "oka@oka.no"}]}';
```

Future Piperpal work

Convolutional Neural Net to group the nearby points.

Plan to do lookups with a Convolution Neural Network [CNN] using a matrix with all of the entries within a radius away from the user’s location for a given time as computed by a convolution matrix for the longitude and latitude.

$$CNN(lat, lon) = \left[\int_{-90}^{90} F(lat) \left[G(v) e^{2\pi i v lat - lat'} dv \right] dlat', \int_{-180}^{180} F(lon) \left[G(v) e^{2\pi i v lon - lon'} dv \right] dlon' \right]$$

Actual implementation will differ from the equation.

References

[CNN] Amani V. Peddada, James Hong: Geo-Location Estimation with Convolutional Neural Networks
http://cs231n.stanford.edu/reports/CS231N_Final_Report_amanivp_jamesh93.pdf