CSCV337 Course Project FindSomeCache - Web-based GeoCache Search

Introduction

The goal of this assignment is to leverage multiple web application technologies to build a dynamic application. This application provides a web--based front-end for a fictional geocache database. A geocache is typically a small container, planted at a location referenced by its **latitude/longitude**. The activity called geocaching is where participants attempt to locate these containers using GPS. The general location of some geocaches are given by latitude/longitude coordinates, but the exact placement of a cache may be discovered by solving a riddle or puzzle. There are tons of these things hidden everywhere and it's a fun way to explore new places. For more information about the activity and to find real geocaches, visit this website: https://www.geocaching.com/play

Requirements

You must provide a web--based interface for visitors to see a listing of geocache locations and allow them to filter the results by cache type, difficulty, or distance from a user--specified latitude/longitude or the user's current location. At the beginning, you may assume the user is at Tucson, Arizona. To represent the user position, you may use a variable ***pos*** and initialize the variable value as below:

```
pos = {lat: 32.253, lng: -110.912};
```

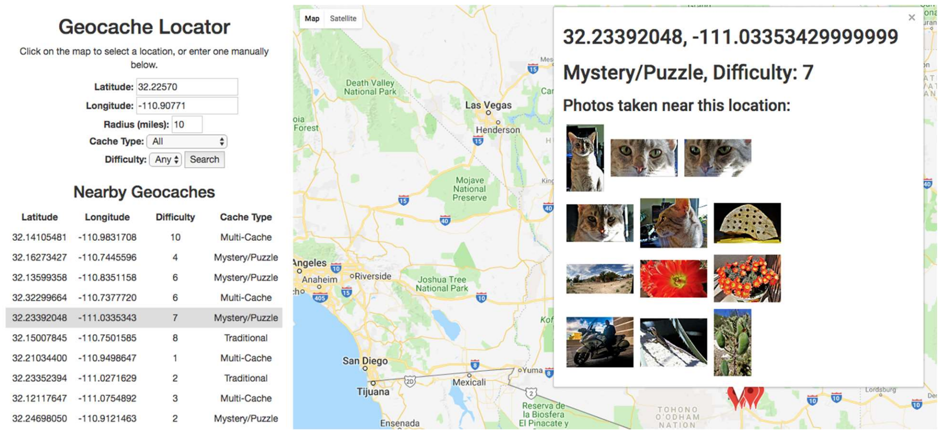
Users should be able to select a distance value as a multiple of 5, with the minimum value set to 5 miles, and the maximum value being 200 miles. **NOTE:** If the user doesn't specify a distance value, your application should default to 10 miles.

When a user selects to filter based on distance from a location, you must center the map widget on the latitude/longitude and build a Bounding Box to filter which geocache records should be displayed. A geocache record should be displayed if the latitude/longitude are within the bounds of the bounding box. NOTE: The Google Maps API has a number of useful JavaScript objects to build a bounding box (or circle) for you. Please refer to the google.maps.Circle and google.maps.Rectangle JavaScript classes for more information. Each of these classes has a

Commented [1]: The bounding box could be implemented by using google.maps.Circle object. Some tip is provided in the Tip section.

getBounds() method to return an object of type google.maps.LatLngBounds which can define the bounding box.

In your user interface, you will display a Google Maps widget that will contain clickable datapoints for each of the geocache listings returned for the user's query. In addition to displaying the geocache data within the map widget, you must also display a tabular listing of the relevant locations. Upon selecting a location (using either the map or table), you must display the information about the selected geocache, and display thumbnails of photos taken near the geocache's latitude/longitude coordinates. To obtain a listing of photos based on a location, you will use the Flickr flickr.photos.search API. You may display this data any way you choose, so long as it is clear to the user that the photos displayed are associated with the selected geocache entry. [Sample screenshot you may want to reference:]



Geocache Locator

Click on the map to select a location, or enter one manually below.

Latitude: 32.22570
Longitude: -110.90771
Radius (miles): 10
Cache Type: All
Difficulty: Any
Search

Nearby Geocaches

Latitude	Longitude	Difficulty	Cache Type
32.14105481	-110.9831708	10	Multi-Cache
32.16273427	-110.7445596	4	Mystery/Puzzle
32.13599358	-110.8351158	6	Mystery/Puzzle
32.32299664	-110.7377720	6	Multi-Cache
32.23392048	-111.0335343	7	Mystery/Puzzle
32.15007845	-110.7501585	8	Traditional
32.21034400	-110.9498647	1	Multi-Cache
32.23352394	-111.0271629	2	Traditional
32.12117647	-111.0754892	3	Multi-Cache
32.24698050	-110.9121463	2	Mystery/Puzzle

32.23392048, -111.03353429999999

Mystery/Puzzle, Difficulty: 7

Photos taken near this location:

Note that for the list of photos searched based on the clicked location from Flickr, the display window only presents no more than 12 thumbnails of photos. You may reference the above page design. But, feel free to be creative!

User Interface

The design for your user interface will not be defined in this specification. You must design a user interface that you feel is functional and intuitive to use. The design of your UI is not a major component for grading. There are no image files associated with this assignment. You may use any framework you wish in order to implement the user interface.

Provided Files and Test Database

The Comma Separated Value (CSV) files provided for this project were imported into a MySQL database named ***test*** for your use.   The ***test*** database contains two tables **test_data** and

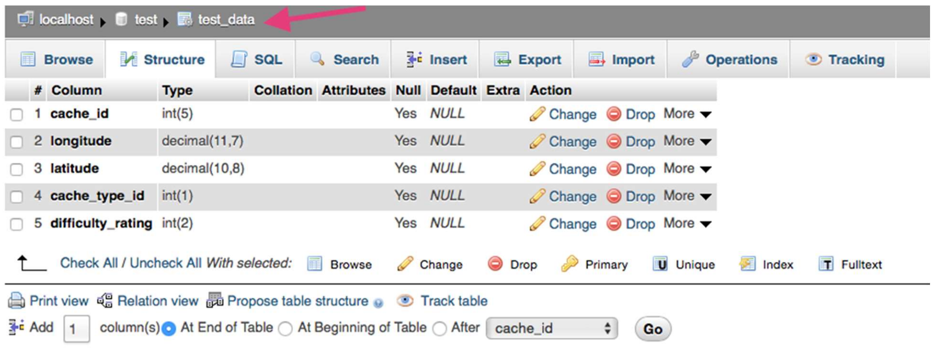
cache_types. The two table data was populated based on the imported data from the two csv files. (You may want to have a check on the data contained in the two csv files.)

test_data.csv

This CSV file contains all the fictional geocache points for our application. Each row in the file represents the data for a single geocache location.

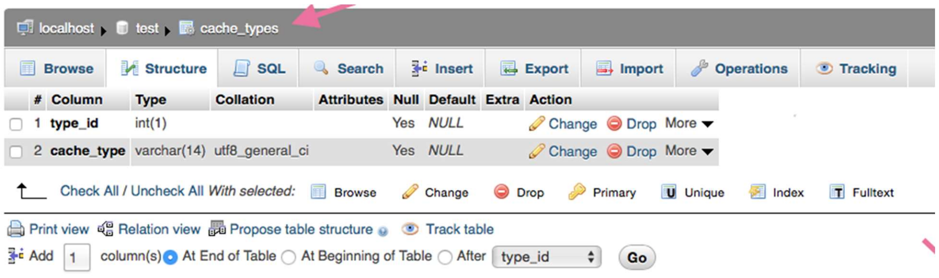
cache_types.csv

This CSV file contains an integer--to--string mapping of a cache type identifier to cache type name. The cache type identifier integer is referenced within the 4th column of the data in test_data.csv.
The two tables have columns with same names as presented on the first lines in the two provided csv files. I provide two screenshots that present the table structures in the ***test*** database. You need to compose your queries based on the table structures.


localhost ▸ test ▸ test_data
Browse Structure SQL Search Insert Export Import Operations Tracking

#	Column	Type	Collation	Attributes	Null	Default	Extra	Action
1	cache_id	int(5)			Yes	NULL		Change Drop More
2	longitude	decimal(11,7)			Yes	NULL		Change Drop More
3	latitude	decimal(10,8)			Yes	NULL		Change Drop More
4	cache_type_id	int(1)			Yes	NULL		Change Drop More
5	difficulty_rating	int(2)			Yes	NULL		Change Drop More

Check All / Uncheck All With selected: Browse Change Drop Primary Unique Index Fulltext
Print view Relation view Propose table structure Track table
Add 1 column(s) At End of Table At Beginning of Table After cache_id Go


localhost ▸ test ▸ cache_types
Browse Structure SQL Search Insert Export Import Operations Tracking

#	Column	Type	Collation	Attributes	Null	Default	Extra	Action
1	type_id	int(1)			Yes	NULL		Change Drop More
2	cache_type	varchar(14)	utf8_general_ci		Yes	NULL		Change Drop More

Check All / Uncheck All With selected: Browse Change Drop Primary Unique Index Fulltext
Print view Relation view Propose table structure Track table
Add 1 column(s) At End of Table At Beginning of Table After type_id Go

Database Connectivity

For this project, we will be using the same MySQL database management system as we used in Assignment 6. To connect to the ***test*** database using PHP, you need to use the same username and password used in Assignment 6 to connect to

Host: 150.135.53.5
Port: 3306
Database Name: **test**

The database is ***test*** this time, not imdb.

External APIs

Google Maps JavaScript API

https://developers.google.com/maps/documentation/javascript/tutorial

In order to use the Google Maps API, you must first obtain an API key for your application. This API key is required for all requests and is used by Google to track how many requests your application has made for the API. Please read the documentation at the Google Maps API link provided above for information about using the API to display a map, manipulate the positioning/zoom factor on the map, and respond to clicks on data points within the map.

NOTE: Using the Google Maps API is free and Google provides up to 25,000 requests per day in their free tier. If in the unlikely event that your application exceeds the quota, your API calls will fail until you enable billing. Same goes for most of Google's other services such as App Engine.

Flickr Photo API

As with the Google Maps API, you must also obtain an API key for the Flickr service (Yahoo! account required). Once you have an API key, you can make HTTP GET requests to the flickr.photos.searchmethod to retrieve photos within a specified bounding box. NOTE: Flickr's bounding box parameter (bbox) requires comma--separated parameters in the following order:

<Southwest long>,<Southwest lat>,<Northeast long>,<Northeast lat>
Once you have the data in the desired format, you can create URLs for the returned images, which can be referenced within your UI. Please use the URL guide as a reference for building these image URLs.

Framework Constraints

If there is a web framework you wish to use such as AngularJS, jQuery, jQuery Mobile, Bootstrap or any other UI or JS framework, please feel free to use it! Have fun with this assignment.

Deploying Your Project

In addition to submitting your project via D2L, you **must** post your project to your protected UA web hosting space. I will test your project code live. Note that. u.arizona.edu is PHP- enabled and has libcurl and MySQL bindings installed for PHP. For information about the PHP installation on u.arizona.edu, please browse to http://u.arizona.edu/~milazzom/phpinfo.php

In order to prevent your site from being indexed by Google, you may use meta tag to instruct Google to block access to your web pages.

Tips to Implement the Project

Google Map API

When a user selects to filter based on distance from a location, you can use use the below code to locate the zone and find the zone boundaries in terms of maximum and minimum latitude/longitude.

```
 //locate the zone using the new position represented by newPos and
 // the distance calculated into meters represented by variable rad
// 1 mile is almost equal to 1609 meters
  var zone = new google.maps.Circle({center: newPos, radius: rad});
  var bounds = zone.getBounds();

  maxLat = bounds.getNorthEast().lat();
  minLat = bounds.getSouthWest().lat();
  maxLng = bounds.getNorthEast().lng();
  minLng = bounds.getSouthWest().lng();
```

JSON PHP

In your php code to retrieve data from the test database, you may use below code to encode the rows from database table into JSON data.

```
$db = new PDO("WHAT EVER TO PUT HERE”...);
$query = “some query statement here”;
$results = $db->query($query);
$rows = $results->fetchAll();
```

```
echo json_encode($rows);
```

At the client side, you may use JavaScript ajax request, assuming held by variable ***xhr***, to get the response data from the server. After the server data is received, you may use below code to parse the response text into an array object or whatever object the JSON data represents:

```
var data = JSON.parse(xhr.responseText);
```

Google Map Marker

You can create a Google Map Marker object to represent a clickable data point on a map based on the a position.

```
 //Note that below code is only providing some idea to use google map marker.
//You should feel free to use your own data structure
//Assume that a position is stored in an object x that contains geocache data including
//    latitude, longitude,cache_type, and difficulty_rating

    marker = new google.maps.Marker({
                    map: map,
                    position: new google.maps.LatLng({
                              lat: parseFloat(x.latitude),
                              lng: parseFloat(x.longitude)
                              }),
                    title: x.cache_type + ", Difficulty: " + x.difficulty_rating
                    });

 //You can add some listener to handle event fired by the marker as below
marker. addListener('click', function_to_be_defined);
```

If later you want to remove the marker from the map, you may use

```
marker.setMap(null);
```

Google Map InfoWindow

To use InfoWindow to display text and photo images, you may reference the below JavaScript code:

First, you can create the InfoWindow as below:

```
 //assume that contentString contains the text and image to display in the info window
var infoWindow = new google.maps.InfoWindow({content: contentString});
```

Later, you can write code to change the infoWindow's content as well as position

```
infoWindow.setPosition(pos);
//pos holds a google.map.LatLng object
infoWindow.setContent(contentString);
//contentString holds a string consists of HTML code to display text and photos
```

To open the infoWindow on the map, you can use the below code

```
    infoWindow.open(map);
   //map holds the google.map.Map object
```

Flickr Service

When using the Flickr service to construct photo url, you could construct each thumbnail image based on the searched photo object's attribute data.   The URL pattern could be below:

https://farm{farm-id}.staticflickr.com/{server-id}/{id}_{secret}_[mstzb].jpg

To find what data the Flickr search service provides, you may provide the following URL in your browser and check what data the server provides:

https://api.flickr.com/services/rest/?api_key=<you-api-key>&method=flickr.photos.search&lat=32.2162358&lon=-110.88261449999999

Note that you need to apply an API key first to use the search service. After you have your API key obtained, you can use the key to fill in the above URL and check the retrieved data. Through the above URL, you can search photos at location with latitude = 32.2162358 and longitude=-110.88261449999999.

For each retrieved photo data, you need to retrieve attribute *farm* for farm-id required to construct the image URL, *server* for server-id, *id* for id, *secret* for secret. Since you are required to display only thumbnail images, you can use t from the last option set [mstzb] since t represents thumbnail.

Example:
https://farm1.staticflickr.com/2/1418878_1e92283336_t.jpg

farm-id: 1
server-id: 2
id: 1418878
secret: 1e92283336
size: t

Turn-in

Create a zip file named <UA Net ID>_final.zip containing all files you created or modified for this assignment and submit it to Final Project drop box on our D2L site. You must make your site accessible from your password protected UA web hosting space and submit the URL in your D2L submission notes. In addition to the URL, please also provide me the password for user ***milazzom*** to access your protected web folder in your submission note.

Rubric

This assignment is worth 100 points, broken down as follows:

- 10 points: User interface allows input to demonstrate functionalities of the application
- 30 points: Upson user input submission, the app displays clickable datapoints for each of the geocache listings returned for the user's query.
- 20 points: In addition to displaying the geocache data within the map widget, the app also display a tabular listing of the relevant locations.
- 15 points: Upon selecting a location on the map, you must display the information about the selected geocache, and display thumbnails of photos taken near the geocache's latitude/longitude coordinates. [If photos, if any, cannot be displayed, 5 points will be deducted.]
- 15 points: Upon selecting a location on the table, you must display the information about the selected geocache, and display thumbnails of photos taken near the geocache's latitude/longitude coordinates. [If photos, if any, cannot be displayed, 5 points will be deducted.]
- 10 points: Code presentation