How To Guide: eBird API

Saturday, August 10, 2013

2:58 PM

Name: Sara Thorup

**What is eBird?**

eBird.org is a website developed by the Cornell's Ornithology lab and the National Audubon Society. eBird has multiple purposes - it allows birdwatchers to enter bird sighting information into the eBird database. The information entered is quality controlled by regional experts. In addition to submitting bird sighting information, eBird allows a user to see bird sighting stats by region. For example, a user could view a map of the bird sightings in their area. eBird's sighting information can be used by conservationists to track changes in the bird population over time.

**What is the purpose of the eBird API?**

The eBird API allows a user to query eBird's database. The queries are returned as an xml document. The eBird API has over a dozen different query types that can be further customized by passing various parameters. For example, a user can query where a certain bird was recently sighted or recent sightings of a specific bird in a particular area.

**How do I use the eBird API?**

After completing the following steps, the user will the able to use the eBird API.

Step 1: Learn how to query the eBird database using a URL

Step 2: Learn how to format the query results using PHP

Step 3: Learn how to insert query results into a webpage using jQuery

Example code will be included at the end of the document.

**Assumptions**

This guide assumes that the reader is familiar enough with HTML, XML, PHP, JavaScript, and jQuery to follow along with the instructions presented here.

**Step 1: How to query the eBird database using a URL**

**Query types**

As previously mentioned, the eBird API provides the user with many different query choices. The query is submitted by altering the parameters in the URL. Once submitted, the query returns an XML page.

**Link to the full list of queries:**

<https://confluence.cornell.edu/display/CLOISAPI/eBird+API+1.1>

This guide will walk through how to query the recent bird sightings by a specific location. In general, the same steps apply to all of the queries; although, some queries may take different parameters or return additional fields.

**Documentation for the query that will be demonstrated:**

<https://confluence.cornell.edu/display/CLOISAPI/eBird-1.1-RecentObservationsInARegion>

Documentation for a query includes:

* + the query description
  + URLs for submitting a query (including example URLs)
  + parameter descriptions
  + result field descriptions
  + example query responses

The documentation provides two examples of the URL used to submit a query - one with only the required parameters specified, and the second with all parameters specified. See below for the full list of parameters.

Machine generated alternative text: Parameter Descriptions
Parameter Name Required Default Value Options Example Value Description
rtype yes country, subnationall. subnational2 subnationall the region type you are interested in
r yes region code corresponding to selected region type- see region code reference
back 14 1 - 30 integer 7 the number of days back to look for observations
maxResults 1 - 10000 integer 10 the maximum number of result rows to return in this request to get all results do not include this parameter
locale en_US Java standard locale codes fr_CA Language/locale of response (when translations are available) See http://javasunconVjavase/6/docs/api/javalutil/LocalehtmL
fmt xml json. xml json format of the response
includeProvisional false true.false false set to true if youd like flagged records that have not yet been reviewed to be included in the results
hotspot false true, false false set to true if results should be limited to sightings at birding hotspots

**Example simple query**

The Recent Observations in a Region query requires that the rtype (region type) and r (region) are specified. For example, if a API user wanted to see the birds recently spotted in the United States, they would need to paste this URL into their browser:

<http://ebird.org/ws1.1/data/obs/region/recent?rtype=country&r=US>

In this example rtype = country, meaning that the query will return country level information. r = US, specifying that the query should return data on the United States.

If the API user wanted to view only the birds in Washington State, they would use this URL:

<http://ebird.org/ws1.1/data/obs/region/recent?rtype=subnational1&r=US-WA>

In this second example rtype = subnational1, which means that it is looking at a level below the country level. In this case r = US-WA which indicates Washington State, United States.

At the time of this writing (8/11/2013) the API can only drill down to the state level (subnational1), in the future it may support county level information

**List of region codes**

<https://confluence.cornell.edu/display/CLOISAPI/eBird-1.1-RegionCodeReference>

**Example fully specified query**

<http://ebird.org/ws1.1/data/obs/region/recent?rtype=subnational1&r=US-WA&back=5&maxResults=500&locale=en_US&fmt=xml&includeProvisional=true>

In this example all of the possible parameters have been specified. Please refer to the parameter list above for details. In summary, this query specifies that at most 500 bird sightings (maxResults) that occurred in the last 5 days (back) in Washington state (r and rtype) should be returned as XML (fmt) even if the sightings have not been reviewed (includeProvidsional). Parameter types are separated by the & symbol.

Not all parameters need to be specified. The user could specify the region type, region, and maxResults while omitting the rest to their default values.

**Example XML returned by a query**

The query documentation lists which fields will be returned as nodes in the XML file. Below are the fields returned by the Recent Observations by Region query:

Machine generated alternative text: Simple Result Fields
Field (JSON) Field (XML) Description
comName corn-name species common name not included in the ‘simple’ detail if a scientific name was specified as an input parameter
sciName sci-name species scientific name- not included in the ‘simple’ detail if a scientific name was specified as an input parameter
obsDt obs-dt observation date formatted according to Iso 8601 (eg ‘YYYY-MM-DD’. or ‘YYYY-MM-DD hh:mm’) Hours and minutes are excluded if the observer did not report an observation time
howMany how-many The number observed Not included if only presence was noted
locID bc-id unique identifier for the location
bocationPrivate location-private ‘true’ if location is not a birding hotspot ‘false’ otherwise
bocName bc-name location name
bat bat latitude of the location
Ing Ing longitude of the location
obsReviewed obs-reviewed ‘true’ if obs has been reviewei ‘false’ otherwise
obsValid obs-valid ‘true’ if obs has been deemed valid by either the automatic filters or a regional reviewer. ‘false’ otherwise

Below is a sample of the XML generated by the query of recently sighted birds in Washington state.

The first part of the XML lists the values of the parameters that were passed. This section is helpful because it shows what default values were used if the URL was not fully specified. The rest of the XML details bird sightings. This tutorial will mainly focus on the <com-name> tag.

Machine generated alternative text: — <response>
— <header>
<locale country’US” language’ en7>
<tirnestamp>20 13-08-11 T02:20:33 .392-04 :OO<timestamp>
— <criteria>
<fmt>xml<fmt>
<include-provisional>false< include-provisional>
<hotspot>false<.hotspot>
<rtvpe>subnational 1 <‘rtype>
<r>USWAc,r>
<back>14<back>
<max-results>1 0000c’max-results>
<header>
— <result>
— <sighting>
<loc-id>L2249650c1oc-id>
<location-private>true<iocation-private>
<obs-dt>2013-08-1O 1 7:30%obs-dt>
<obs-reviewed>falsecobs-reviewed>
<ob s-va lid>tnie< obs-va lid>
<loc-narne>My Seattle Yard<loc-name>
<how-many> 1 <how-many>
<lat>47.6554164clat>
<Ing>- 122.3 525959<lng>
<com-name>Bewick’s Wren< corn-name>
<sci-narne>Thryomanes bewickii</sci-narne>
<sighting>

eBird can always return results in XML and can sometimes return results as Json or CSV. Check the specific query's documentation for details.

**Step 1 summary**

In this section the reader learned where to view the list of eBird queries, how to modify the parameters of a query URL, and how to look at the generated XML.

To practice what was demonstrated in Step 1, modify the Recent Observations in a Region query URL so that that the last 5 birds recently spotted in Oregon are displayed.

*Hint: Modify the maxResults and r parameters*

**Step 2: How to format the query results using PHP**

In general, a webpage browser cannot access any document outside of its domain (including XML documents) using AJAX techniques. This means that it is not possible to access the eBird API via AJAX from a third party website domain.

To circumvent this problem, the third party domain can set up a resource within their domain to access the eBird database and parse and format the appropriate data via a server side script. The third party website can then use AJAX to pull data from this resource which in turn pulls data from the eBird API. A PHP script will be used as the intermediary between the eBird API and the third party website.

This step will walk through how to parse an XML file hosted on a different server using a PHP server side script. Step 3 will demonstrate how display the resulting contents of this PHP script in an HTML file.

**Creating the PHP file**

Create a PHP file with the following contents:

Machine generated alternative text: -<?php
ini_set ( ‘ display_errors ‘ , ‘ on, ) ; I/displays errors
I/Loads the XML file as eBirdQuery
$eBirdQuery =
I/For each sighting node in the eBird XML file, display the value of the common name tag
-foreach ($eBirdQuery->result->sighting as $sighting)
echo $sighting->{ ‘coin-name’ };
echo “<br>” ;
I
?>

This PHP script downloads and parses out the content of the XML file specified in the URL using simpleXML. In the foreach loop, specify the path to the element in which the desired information is nested. For example, in the XML above, the com-name element is nested inside the sightings element. The path to the sighting element is saved as $sighting. The loop iterates through each sighting section and pulls out the desired tag, in this case, com-name.

Element names with periods or dashes need to be enclosed in curly braces and quotation marks. Otherwise they can be accessed like $sighting->lat (would get the sighting's latitude value).

**About SimpleXML**

There are many different useful SimpleXML functions. For example the following would display the number of bird sightings:

echo $eBirdQuery->result->sighting->count();

SimpleXML could also be used to display the 10th bird in the list:

echo $eBirdQuery->result->sighting[9]->{'com-name'};

**SimpleXML documentation and examples**

<http://php.net/manual/en/class.simplexmlelement.php>

<http://www.w3schools.com/php/php_ref_simplexml.asp>

**Another example**

Here is another example using more than one element. HTML tags have been added to assist with formatting.

Machine generated alternative text: - <?php
ini_set ( ‘display_errors’ , ‘Qnt ) ; I/displays errors
//Loads the XML file as eßirdQuery
$eBirdQuery = simplexml_load_file (“http://ebird.org/wsl. 1/data/obs/region/recent?rtype=subnationall&r=US—WA”)
I/Start table formatting
echo “<table> \n”
echo “<tr>-<td><b>Common Næne</b>-’</td><td><b>Scientific Name</b></td></tr> \n”;
I/For each sighting node in the eBird XML file, display the value of the common name tag
-foreach ($eBirdQuery->result->sighting as $sighting)
echo “<tr><td>”;
echo $sighting->{ corn-name’);
echo “</td><td>”;
echo $ sighting-> { ‘sci-name’);
echo “</td></tr> \n”;
I/End table form
echo “</table>”;
?>

Open the PHP page to make sure that no errors have occurred and that everything was formatted correctly.

**Step 2 summary**

In this section the reader learned how to load an XML file from one domain with a server side PHP script on a separate domain. The XML is then parsed by the same script to extract the relevant information.

To practice what was demonstrated in Step 2, modify the code sample above so that the bird's common name and location spotted are displayed in a table.

*Hint: Use $sighting->{'loc-name'}*

**Step 3: How to insert query results into a webpage**

It is possible to only have a PHP page for the user to view the query results; however, looking forward to more complicated pages, it is probably better to have a separate HTML page for the user to view query results. This step will go over how to insert the data from the PHP page into an HTML page using jQuery.

Create a HTML page in the same location as the PHP page. Include the following information:

Machine generated alternative text: 1 <!ÐOCTYPE html>
2 —<html>
3 —<head>
4 <mcta http—equiv”content-type’1 content’1text/html; ç=UTF-8”>
S <title>eßird Demo</title>
6 <script src”http://aiax.googleapis.com/aiax/libs/iauery/1.lO.2/icruery.min.is”)-</script>
7
— <script>
11 — $(dccument).ready(function() (
12 $(“#resultDescrip”) .hidefl;
13 $(“$hideBirds”) .hidefl;
14
15 — $r#qetßirds9.click(function() (
16 $(“#resultDescrip”) .shcwfl;
17 $(“#results”) .shcwfl;
18 $(“#results”).load(’h:tc:. :z:rtate.edu/—thorups/cs494/howTcGiide/bzr:iEetter.chp’);
19 $(“*hideBirds”) .shcwfl;
20 S(”#getßirds”) .hidefl;
21 ))‘;
22
23 — $(“$hideBirda”).click(fimction() (
24 $(“$hideBirds”) .hidefl;
25 $(“fgetBirds”) .shcwfl;
26 $r#results”) .hidefl;
27 S(”#resultDescrip”) .hidefl;
28
29 ));
30 ));
31 </script>

Machine generated alternative text: 33 </head>
ci
3E -<body>
<h2>eßird Demo</h2>
: This simple HTML page demonstrates how eßird results can be displayed via a PUP page.
<br><br>
- <button id”getBirds”>Get Results</button>
42 <button id “hideBirds”Æide Results</button>
43
44 <br)-<br>
45 <div id = “resultÐescrip”)-<h2>Recent bird sightings in Washington State</h2>-</div>
46 <div id”results”>eßird results will show up here</div>
47
4E </body>
49 </html>

The key here is that when the button is clicked the PHP output is loaded into the contents of the specified <div> tag using the load() function.

$("#results").load('http://web.engr.oregonstate.edu~thorups/cs494/howToGuide/birdGetter.php');

**Info on the .load() function**

<http://api.jquery.com/load/>

**Step 3 summary**

In this section the reader learned how to load the output of a PHP script (containing eBird data) into their HTML page using the .load() function.

To practice what was demonstrated in Step 3, consider making different PHP files for multiple query types, then ask the user to choose which one to display using buttons.

**Going forward**

In this tutorial, the reader learned how to query the eBird database by modifying the query URL. The reader then created a PHP script to parse the XML generated by eBird. The PHP page was then loaded into an HTML page to be viewed.

With this information more complicated projects could be made. For example, the user could select the country or state from which to view data via a dropdown. Or perhaps the page could display a map of where birds were recently spotted. Or a message could be sent to a user when a specific bird was spotted in their area.

**Resources (by topic)**

**eBird**

eBird: [www.eBird.org](http://www.eBird.org)

eBird API: <https://confluence.cornell.edu/display/CLOISAPI/eBird+API+1.1>

Region codes: <https://confluence.cornell.edu/display/CLOISAPI/eBird-1.1-RegionCodeReference>

**simpleXML**

Documentation: <http://php.net/manual/en/class.simplexmlelement.php>

Tutorial: <http://www.w3schools.com/php/php_ref_simplexml.asp>

**.load()**

Documentation: <http://api.jquery.com/load/>

**Example Code**

**PHP example with table formatting**

<?php

ini\_set('display\_errors', 'On'); //displays errors

//Loads the XML file as eBirdQuery

$eBirdQuery = simplexml\_load\_file("http://ebird.org/ws1.1/data/obs/region/recent?rtype=subnational1&r=US-WA");

//Start table formatting

echo "<table> \n";

echo "<tr><td><b>Common Name</b></td><td><b>Scientific Name</b></td></tr> \n";

//For each sighting node in the eBird XML file, display the value of the common name tag

foreach ($eBirdQuery->result->sighting as $sighting) {

echo "<tr><td>";

echo $sighting->{'com-name'};

echo "</td><td>";

echo $sighting->{'sci-name'};

echo "</td></tr> \n";

}

//End table formatting

echo "</table>";

?>

**HTML Page**

<!DOCTYPE html>

<html>

<head>

<meta http-equiv="content-type" content="text/html; charset=UTF-8">

<title>eBird Demo</title>

<script src="http://ajax.googleapis.com/ajax/libs/jquery/1.10.2/jquery.min.js"></script>

<script>

$(document).ready(function() {

$("#resultDescrip").hide();

$("#hideBirds").hide();

$("#getBirds").click(function() {

$("#resultDescrip").show();

$("#results").show();

$("#results").load('http://web.engr.oregonstate.edu/~thorups/cs494/howToGuide/birdGetter.php');

$("#hideBirds").show();

$("#getBirds").hide();

});

$("#hideBirds").click(function() {

$("#hideBirds").hide();

$("#getBirds").show();

$("#results").hide();

$("#resultDescrip").hide();

});

});

</script>

</head>

<body>

<h2>eBird Demo</h2>

This simple HTML page demonstrates how eBird results can be displayed via a PHP page.

<br><br>

<button id="getBirds">Get Results</button>

<button id ="hideBirds">Hide Results</button>

<br><br>

<div id = "resultDescrip"><h2>Recent bird sightings in Washington State</h2></div>

<div id="results">eBird results will show up here</div>

</body>

</html>