

Homework 6: Server-side Scripting

1. Objectives

- Get experience with PHP programming language
- Get experience with Yahoo! GeoPlanet and Weather APIs
- Get experience using an XML parser in PHP.

2. Description

In this exercise, you are asked to create a webpage that allows you to search for weather information using the Yahoo! GeoPlanet and Yahoo! Weather RSS feed APIs, and the results will be displayed in tabular format.

A user will first open a page, called **get_weather.php (or any valid page name)**, where he/she can enter a City or ZIP Code, select the corresponding type and the temperature unit (Fahrenheit, Celsius). An example is shown in Figure 1.

Weather Search

Location:	<input type="text"/>	
Location Type:	<div>City ▼</div>	
Temperature Unit:	<div>City ZIP Code</div>	<input type="radio"/> Celsius
<input type="button" value="Search"/>		

Figure 1: Initial Screen to Enter the Location and Select the Type

The drop-down list needs to contain the two options shown. Make “City” to be default-selected. After the user has entered the location and type and then clicks the “Search” button, your program will check if the user has entered a location. If the user did not enter anything for location and clicked on “Search”, then an alert message should be shown with an appropriate message prompting the user for valid input. An example of the alert message is shown in Figure 2, and an example of valid input is shown in Figure 3.

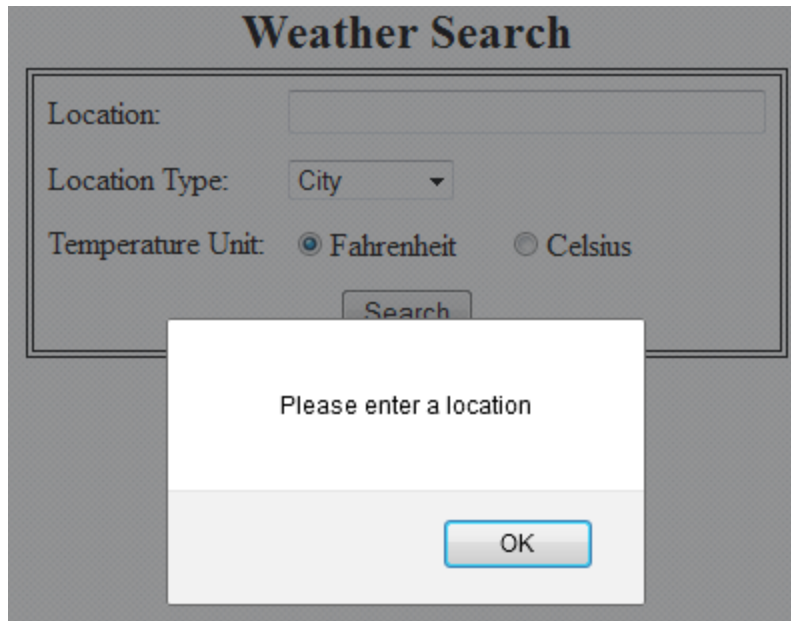


Figure 2: Alert Message

A screenshot of the "Weather Search" form. The "Location:" text box contains the word "Pasadena". The "Location Type:" dropdown menu is set to "City". The "Temperature Unit:" radio buttons show "Celsius" as the selected option. The "Search" button is visible at the bottom of the form.

Figure 3: Enter Location, Select Type and Click “Search” Button

When the input is valid, clicking on the Search button will send a request to your server for **get_weather.php (or whatever your valid page name is)** with the form data (You can use either GET or POST in the form action). This script will grab the data sent to it and send the search information to GeoPlanet (<http://developer.yahoo.com/geo/geoplanet/>). The PHP script will use the data (location and type) to construct a web service URL to query the GeoPlanet API appropriately:

- If the user enters “90007” and chooses the “Zip Code” type, then the URL would look like this:
http://where.yahooapis.com/v1/concordance/usps/90007?appid=<your_app_id>
- If the user enters “Pasadena” and chooses the “city” type, then the URL would look

like this:

[http://where.yahooapis.com/v1/places\\$and\(.q\('Pasadena'\),.type\(7\)\);start=0;count=5?appid=<your_yahoo_app_id>](http://where.yahooapis.com/v1/places$and(.q('Pasadena'),.type(7));start=0;count=5?appid=<your_yahoo_app_id>)

Notes:

- The value of the “type” parameter is 7 to instruct the GeoPlanet API that the results should be only cities.
- To get only the first five places matching the search criteria, “start” and “count” parameters are set to Zero and Five (start = 0, count = 5).
- USPS parameter is used to query about the ZIP codes defined by the US Postal Services. An alternative is CAPS which is used to look for postal codes defined by Canada Post but in this homework we are only concerned of USPS codes.
- Getting a Yahoo application ID is explained in section 3.1

For more detailed information about constructing the URL, read the API

<http://developer.yahoo.com/geo/geoplanet/guide/api-reference.html>.

One key piece of data returned is the **Where on Earth Identifier (WOEID)**, which is a unique 32 bit reference identifier assigned by Yahoo! to identify any feature on Earth. You will see in the next step that you must send a WOEID to the Yahoo! Weather RSS feed to get the weather for a location.

Your PHP script will need to parse the XML file returned from GeoPlanet to extract WOEID:

- For ZIP Code results, your XML parser should read the value of WOEID tag. Figure 4 shows the returned XML from GeoPlant API when querying about the Zip Code “90007”.

```
-<concordance yahoo:uri="http://where.yahooapis.com/v1/concordance/usps/90007" xml:lang="en-US">  
  <woeid>12795615</woeid>  
  <usps>90007</usps>  
</concordance>
```

Figure 4: Example of GeoPlanet XML File When Querying About ZipCode

- For city results, the returned XML file may contain multiple WOEIDs for different cities. Figure 5 shows a part of the retrieved XML file from GeoPlanet API when querying about “Pasadena” city. In this case, the XML file contains a set of <place> tags. In each Place tag there are many sub-tags and one of them is WOEID which is required to extract its value.

```

- <places yahoo:start="0" yahoo:count="5" yahoo:total="6">
- <place yahoo:uri="http://where.yahooapis.com/v1/place/2468964" xml:lang="en-US">
  <woeid>2468964</woeid>
  <placeTypeName code="7">Town</placeTypeName>
  <name>Pasadena</name>
  <country type="Country" code="US" woeid="23424977">United States</country>
  <admin1 type="State" code="US-CA" woeid="2347563">California</admin1>
  <admin2 type="County" code="" woeid="12587688">Los Angeles</admin2>
  <admin3/>
  <locality1 type="Town" woeid="2468964">Pasadena</locality1>
  <locality2/>
  <postal/>
+ <centroid></centroid>
+ <boundingBox></boundingBox>
  <areaRank>4</areaRank>
  <popRank>10</popRank>
</place>
- <place yahoo:uri="http://where.yahooapis.com/v1/place/2468963" xml:lang="en-US">
  <woeid>2468963</woeid>
  <placeTypeName code="7">Town</placeTypeName>
  <name>Pasadena</name>
  <country type="Country" code="US" woeid="23424977">United States</country>
  <admin1 type="State" code="US-TX" woeid="2347602">Texas</admin1>
  <admin2 type="County" code="" woeid="12590107">Harris</admin2>
  <admin3/>
  <locality1 type="Town" woeid="2468963">Pasadena</locality1>
  <locality2/>
  <postal/>
+ <centroid></centroid>
+ <boundingBox></boundingBox>

```

Figure 5: Example of GeoPlanet XML File When Querying About City

Next, using the WOEID value, another web service URL needs to be constructed to query the Yahoo! Weather RSS feed to get the weather for each location, such as

<http://weather.yahooapis.com/forecastrss?w=2468964&u=c>

Notes:

- There are two parameters. W is to set WOEID value. U is to specify the temperature unit celsius or fahrenheit. If temperature in celsius so u=c and if temperature in fahrenheit so u=f.
- For more details about Yahoo Weather RSS you can visit this web page: <http://developer.yahoo.com/weather/>.

A sample result of Yahoo! Weather RSS feed is shown in Figure 6. The following data needs to be extracted:

1. The URL for the Weather Icon
2. The text of the weather condition

3. The temperature
4. The temperature Unit (C or F)
5. Location (city, region, country)
6. Geographical-Location info (latitude and longitude)
7. The URL of full forecast at Yahoo Weather

```

<?xml version="1.0" encoding="UTF-8" standalone="no" xmlns:geo="http://www.w3.org/2003/01/geo/wgs84_pos#" xmlns:yweather="http://xml.weather.yahoo.com/ns/rss/1.0" version="2.0">
  <channel>
    <title>Yahoo! Weather - Pasadena, CA</title>
    <link>
      http://us.rd.yahoo.com/dailynews/rss/weather/Pasadena_CA/*http://weather.yahoo.com/forecast/USCA0840_c.html
    </link>
    <description>Yahoo! Weather for Pasadena, CA</description>
    <language>en-us</language>
    <lastBuildDate>Tue, 24 Sep 2013 3:52 pm PDT</lastBuildDate>
    <ttl>60</ttl>
    <yweather:location city="Pasadena" region="CA" country="United States"/>
    <yweather:units temperature="C" distance="km" pressure="mb" speed="km/h"/>
    <yweather:wind chill="32" direction="120" speed="12.87"/>
    <yweather:atmosphere humidity="16" visibility="16.09" pressure="1009.4" rising="2"/>
    <yweather:astronomy sunrise="6:42 am" sunset="6:47 pm"/>
    <image>
      <title>Yahoo! Weather</title>
      <width>142</width>
      <height>18</height>
      <link>http://weather.yahoo.com</link>
    </image>
    <url>
      http://l.yimg.com/a/i/brand/purplelogo/uh/us/news-wea.gif
    </url>
    <item>
      <title>Conditions for Pasadena, CA at 3:52 pm PDT</title>
      <geo:lat>34.15</geo:lat>
      <geo:long>-118.14</geo:long>
      <link>
        http://us.rd.yahoo.com/dailynews/rss/weather/Pasadena_CA/*http://weather.yahoo.com/forecast/USCA0840_c.html
      </link>
      <pubDate>Tue, 24 Sep 2013 3:52 pm PDT</pubDate>
      <yweather:condition text="Fair" code="34" temp="32" date="Tue, 24 Sep 2013 3:52 pm PDT"/>
      <description>
        <![CDATA[
          <br />
          <b>Current Conditions:</b><br />
          Fair, 32 C<br />
          <br />
          <b>Forecast:</b><br />
          Tue - Clear. High: 31 Low: 16<br />
          Wed - Sunny. High: 27 Low: 14<br />
          Thu - Sunny. High: 26 Low: 13<br />
          Fri - AM Clouds/PM Sun. High: 29 Low: 15<br />
          Sat - Sunny. High: 32 Low: 16<br />
          <br />
          <a href="http://us.rd.yahoo.com/dailynews/rss/weather/Pasadena_CA/*http://weather.yahoo.com/forecast/USCA0840_c.html">Full Forecast at Yahoo! Weather</a><br />
          (provided by <a href="http://www.weather.com">The Weather Channel</a><br />
        ]]>
      </description>
      <yweather:forecast day="Tue" date="24 Sep 2013" low="16" high="31" text="Clear" code="31"/>
      <yweather:forecast day="Wed" date="25 Sep 2013" low="14" high="27" text="Sunny" code="32"/>
      <yweather:forecast day="Thu" date="26 Sep 2013" low="13" high="26" text="Sunny" code="32"/>
      <yweather:forecast day="Fri" date="27 Sep 2013" low="15" high="29" text="AM Clouds/PM Sun" code="30"/>
      <yweather:forecast day="Sat" date="28 Sep 2013" low="16" high="32" text="Sunny" code="32"/>
      <guid isPermaLink="false">USCA0840_2013_09_28_7_00_PDT</guid>
    </item>
  </channel>
</rss>
<!--
api20.weather.gql.yahoo.com Tue Sep 24 23:27:29 PST 2013
-->

```

Figure 6: Example of Yahoo! Weather RSS Feed

After extracting the data the PHP script should display data in a tabular format, including how many results were found, below the search form. A sample output is shown in Figure 7.

5 result(s) for City Pasadena






Weather	Temperature	City	Region	Country	Latitude	Longitude	Link to Details
	Fair 19° C	Pasadena	CA	United States	34.15	-118.14	Details
	Fair 22° C	Pasadena	TX	United States	29.69	-95.2	Details
	Partly Cloudy 8° C	Pasadena	MD	United States	39.11	-76.57	Details
	Light Rain 24° C	Pasadena	FL	United States	27.77	-82.73	Details
	Light Drizzle 9° C	Pasadena	NL	Canada	49.02	-57.58	Details

Figure 7: Search Result

You need to map the data extracted from Yahoo Weather RSS feed to your table output as the following:

Table Column	Data From Yahoo Weather RSS Feed
Weather	<ul style="list-style-type: none"> image source: It is embedded in value of <code><description></code> tag. The character data "CDATA" contains <code></code> tag. The source of this image tag should be extracted. (Hint: your XML parse should read the value <code><description></code> tag then you may use a regular expression to extract only the text quoted in src attribute) Alternate Text and Title: value of <code>"text"</code> attribute of <code><yweather:condition></code> tag
Temperature	<ul style="list-style-type: none"> Temperature Condition: value of <code>"text"</code> attribute of <code><yweather:condition></code> tag Temperature: value of <code>"temp"</code> attribute of <code><yweather:condition></code> tag Temperature Unit: value of <code>"temperature"</code> attribute of

	<yweather:units> tag
City	value of “city” attribute of <yweather:location> tag
Region	value of “region” attribute of <yweather:location> tag
Country	value of “country” attribute of <yweather:location> tag
Latitude	value of <geo:lat> tag
Longitude	value of <geo:long> tag
Link Details	value of <link> which is a child of <channel> tag

If a search result is missing any of the data listed above, display N/A for the data that was not found.

The Weather Icon must be displayed, and the alt text and title must be the text of the weather condition. Clicking on the image will open the Weather Feed page in a new window or tab. Clicking on “Details” will open the full Yahoo! Weather service in a new window or tab.

If a search is entered for which there is no result from GeoPlanet API, for example, a search about “AAAA” city, instead of the table, a message (not a popup) should be displayed to show that there is no result for the query. See Figure 6.



Zero results found!

Figure 8: No Result

There is another case. When querying Yahoo Weather RSS feed about a specific WOEID, the RSS feed does not provide weather information. In this case the table row corresponding to this WOEID should be discarded. For example, searching GeoPlanet for “San Francisco” city returned 5 places named “San Francisco” with different WOEIDs (2487956, 20228461, 23414844, 346589, 346588). Two of these WOEIDs (20228461,

346588) do not return weather information when querying Yahoo! Weather RSS feed so their rows are not displayed in the result table as shown in Figure 9.

3 result(s) for City San Francisco




Weather	Temperature	City	Region	Country	Latitude	Longitude	Link to Details
	Fair 61° F	San Francisco	CA	United States	37.78	-122.42	Details
	Partly Cloudy 70° F	San Francisco	N/A	Spain	37.42	-1.88	Details
	Rain 63° F	San Francisco	N/A	Spain	42.76	-9.07	Details

Figure 9: Results When Searching About San Francisco

In summary, the general mechanism which should be implemented is as the following:

- Based on the chosen search criteria, construct a proper web service URL to retrieve the XML file from GeoPlanet API.
- Parse the returned XML and extract WOEID value(s).
- Call Yahoo! Weather RSS feed (WOEID is a parameter in the web service URL) and retrieve XML file.
- Parse the returned XML file and extract: geolocation, location (city/region/country), weather (temperature, condition, icon).

3. Hints


3.1. How to get Yahoo Application ID

Go to <http://developer.yahoo.com/geo/geoplanet/>. In “How Do I Get Started?” section click on “Application ID”. You need to log in using your Yahoo ID. After that you should fill out a form as shown in Figure 9. Choose Generic authentication method, fill the other necessary fields, and press on “Continue” button. Then your Yahoo Application ID is given. Copy this ID which you will use in Yahoo GeoPlanet webserive URL.

We need some information from you...

To use Yahoo! Web Services, we need some information about you and the application you're building. We collect this information to get a better understanding of how Yahoo! Web Services are being used and to protect the security and privacy of Yahoo! users.

If you've already registered for an application ID, [you can see them here](#).


Developer Registration

Fields marked with an asterisk * are required.

*Yahoo ID:

*Authentication method:

Click [here](#) for more information

☒ **Generic, No user authentication required**
This appid will allow you to make calls to our non-authenticated web services

☐ **Browser Based Authentication**
Use this option for browser applications

*Developer/Company Name:

For example: 'Joe/Jane Developer' or 'BigCo Inc.'

*Product name:

For example: 'My Yahoo! Enabled Web App'

Web Application URL:

For example: 'http://myapp.com/welcome.html'

*Contact email:

For example: 'developer@domain.com'

Phone number:

For example: '123-456-7890'

*Description of application:

Figure 9: Yahoo Application ID Form

3.2. Parsing XML files in PHP

You are free to choose any XML parsing library but we recommend *SimpleXML* API. The SimpleXML library is a simple way of getting an XML element's name, attributes, and text. As of PHP 5, the SimpleXML functions are part of the PHP core. No installation is required to use these functions. The following two tables show a set of functions which you may use. For more detailed information, please read:

- http://www.w3schools.com/php/php_xml_simplexml.asp
- <http://php.net/manual/en/book.simplexml.php>
- http://www.w3schools.com/php/php_ref_simplexml.asp

PHP 5 SimpleXML Functions

Function	Description
__construct()	Creates a new SimpleXMLElement object
addAttribute()	Adds an attribute to the SimpleXML element
addChild()	Adds a child element the SimpleXML element
asXML()	Formats the SimpleXML object's data in XML (version 1.0)
attributes()	Returns attributes and values within an XML tag
children()	Finds the children of a specified node
count()	Counts the children of a specified node
getDocNamespaces()	Returns the namespaces DECLARED in document
getName()	Returns the name of the XML tag referenced by the SimpleXML element
getNamespaces()	Returns the namespaces USED in document
registerXPathNamespace()	Creates a namespace context for the next XPath query
saveXML()	Alias of asXML()
simplexml_import_dom()	Returns a SimpleXMLElement object from a DOM node
simplexml_load_file()	Converts an XML file into a SimpleXMLElement object
simplexml_load_string()	Converts an XML string into a SimpleXMLElement object
xpath()	Runs an XPath query on XML data

PHP 5 SimpleXML Iteration Functions

Function	Description
current()	Returns the current element
getChildren()	Returns the child elements of the current element
hasChildren()	Checks whether the current element has children
key()	Return the current key

next()	Moves to the next element
rewind()	Rewind to the first element
valid()	Check whether the current element is valid

4. Files to submit

On your course homework page, you should update HW6 link to refer your new PHP page. Also, Submit your files (likely only a .php file) electronically to the csci571 account so that they can be graded and compared to all other students' code via the MOSS code comparison tool. Also, you need to submit a README file that briefly describes the details of the organization of your program and how it works.