Session 3: Writing Web Service Clients

Before you start - modify your web service to feature LIKE

Change the SELECT statement from your web service from last week to read:

```
$result=$conn->query("SELECT * FROM wadsongs WHERE artist LIKE '%$a%'");
```

LIKE queries match part of a string, so that if an artist has a joint credit, e.g. "Queen and David Bowie" (Under Pressure), a search for either artist will match it (in this example, searches for both Queen and David Bowie should find this song).

Introduction

So far, we have looked at JSON web services. In this topic we will be looking at how we can connect to a remote web service (provided by someone else) from our own website. A real-world example of this might be an airline booking site such as Expedia contacting web services provided by different airlines, parsing the JSON returned and integrating the data in its own website.

Or, another (as we discussed last week) might be a "fan" website for a particular artist, which would like to use our HitTastic! JSON web service (from last week) to list all the songs by that artist.

There are two questions we need to ask here:

1. How do we connect to another website from a server-side script?
2. How do we parse (interpret) the JSON returned by the web service?

Network Programming with cURL

Every contemporary programming language typically features a library which allows you to connect to a remote server over the network from a client application. Such a client application might be a Java program making connections to a remote server, or alternatively a server-side script running on a given server which wishes to connect to another server.

cURL is one such library used to connect to remote servers, and is the standard way of communicating with a server from PHP. Here is a commented example.

```
// Initialise the cURL connection
$connection = curl_init();

// Specify the URL to connect to
curl_setopt($connection, CURLOPT_URL, "http://remoteserver/script.php");


// This option ensures that the HTTP response is *returned* from curl_exec(),
// (see below) rather than being output to screen.
curl_setopt($connection,CURLOPT_RETURNTRANSFER,1);

// Do not include the HTTP header in the response.
curl_setopt($connection,CURLOPT_HEADER, 0);

// Actually connect to the remote URL. The response is
// returned from curl_exec() and placed in $response.
$response = curl_exec($connection);

// Close the connection.
curl_close($connection);
```

This code makes a connection to a given remote server (here, http://remoteserver/script.php) and the response sent back is stored in $response. If the remote script sends back JSON, $response will contain JSON. If the remote script sends back an HTML page, $response will contain HTML. This is standard code that can be copied and pasted every time you want to make a remote connection just change the URL.

JSON parsing

Having obtained the JSON returned from the server, it is stored in $response. The next thing we need to do is parse (interpret) it.

JSON parsing from PHP is straightforward. You use ***json_decode()***, which performs the opposite function to ***json_encode()***. It loads JSON into memory as PHP arrays, or PHP associative arrays.

Here is an example:

```
$data = json_decode($json, true);
```

(the second parameter, "true", loads the json into a PHP associative array). Note that json_decode() returns null if there is an error with the JSON. The web service should relay the success, or otherwise, of json_decode() to the client, remembering that the client could be any piece of software, not necessarily a browser. Here is a full example of using json_decode() to parse some JSON (the presidential candidates example, above):

```
$json = '[
    {"name" : "Barack Obama", "age" : 55, "nationality" : "US", "job": "President"},
    {"name" : "Hillary Clinton", "age" : 68, "nationality" : "US",
     "job": "Presidential candidate" }
]';
$data = json_decode($json, true);
for($i=0; $i<count($data); $i++)
{
    echo "Name " . $data[$i]["name"] . " " .
         "Age " . $data[$i]["age"] . " " .
         "Nationality " . $data[$i]["nationality"] . " " .
         "Job " . $data[$i]["job"] . "<br/>";
}
```

This example stores some JSON (an array of JSON objects representing US presidents/presidential candidates) in a variable ***$json*** and then converts it to a PHP array of associative arrays, one associative array per person.

Alternative - PHP objects

If you are familiar with object-oriented PHP: note the ***true*** as the second parameter to json_decode. This indicates we are converting the JSON to associative arrays. If we missed out the ***true*** the JSON would be converted into an array of PHP objects instead, so we would need code like this to read the data:

```
$json = '[
    {"name" : "Barack Obama", "age" : 55, "nationality" : "US", "job": "President"},
    {"name" : "Hillary Clinton", "age" : 68, "nationality" : "US",
     "job": "Presidential candidate" }
]';
$data = json_decode($json);
for($i=0; $i<count($data); $i++)
{
    echo "Name " . $data[$i]->name . " " .
         "Age " . $data[$i]->age . " " .
         "Nationality " . $data[$i]->nationality . " " .
         "Job " . $data[$i]->job . "<br/>";
}
```

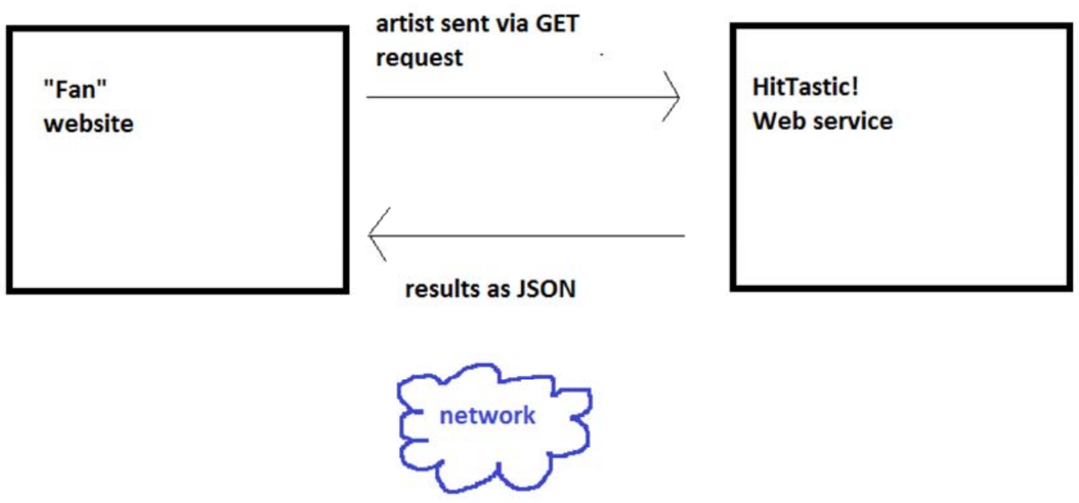
Standard Exercise

The idea of this exercise is to write a "fan" website for your favourite artist (as long as they have had a UK number one hit). Your "fan" website should include a short biography of the artist (get this from Wikipedia if necessary) and should ideally have some CSS appropriate for that artist: maybe black background and white text for an indie or rock band, for example. Just use a famous artist like Oasis, Madonna or the Beatles if you are not sure which artist to use.

The "fan" website would like to list all the number one hits by that artist. However, rather than storing it in its own database, it would like to use the HitTastic! JSON web service that you wrote last week.

Add PHP code on the "fan" website to connect to your web service using cURL and parse the JSON using ***json_decode ()***.

This scenario is illustrated by the diagram below:



To illustrate the fact that the "fan" website is a ***separate*** website to HitTastic!, you will write your "fan" site on the **Edward** server (same username and password as Edward2) and connect to your HitTastic! web service on Edward2.

To do this:

- Use the cURL code above to connect to your web service. Note that you need the full host name, ***http://edward2.solent.ac.uk***, in your cURL connection. Also note that ***any spaces in the artist name need to be encoded as plus signs (e.g. The+Beatles)*** as otherwise the space will be treated as the marker for the end of the artist name.
- Use ***json_decode()*** to decode the JSON and display the list of all the number ones by that artist. Ideally format it nicely (e.g. in a table).