Session 2: JSON Web Services

Review Quiz

To ensure that you have understood this week's material, please ensure that you complete this Review Quiz at the start of the class.

Look at the PHP code below:

```
$a = $_GET["destination"];
$conn = new PDO("mysql:host=localhost;dbname=flightsdb'
$result = $conn->query("SELECT * FROM flights WHERE des
$allRes = $result->fetch(PDO::FETCH_ASSOC);

foreach($allRes as $curRes)
{
    echo "Date: $curRes[date] ";
    echo "Depart Time: $curRes[time] <br />";
}
```

1. Match the variables to the variable types.

   Variable types:

   - Array of associative arrays
   - Input from a Query String
   - Associative array

   Variables:

   - $a
   - $allRes
   - $curRes

2. What would

   ```
   $allRes[3]["arrivalTime"]
   ```

   represent?
3. Match the variable types to the definitions.

   Variable types:

   - Array
   - Associative array

Definitions:

- A variable holding multiple values, where the values are indexed using non-numerical values (e.g. text)
- A variable holding multiple values, where the values are indexed using numerical values, starting at 0

Is HTML always the best output format?

Think about the revision exercise you did in the first session, where you wrote a PHP script to search the HitTastic! database for all songs by a particular artist. This script displayed all songs by a particular artist as HTML, in HitTastic!'s own styling.

Now imagine this. You want to create a website devoted to a particular artist such as Oasis, David Bowie or the Beatles, including a list of all their songs and when they were in the charts. Rather than storing the information yourself, you want to re-use the information from HitTastic!

This artist website might have its own styling. Could you just put the HTML from HitTastic! directly inside the artist website? Why might this not be the best idea?

Can you think of any other reasons why HTML is not the best format to output from a PHP script?

Once we have answered these questions we will quickly revise ***web services*** and look at ***JSON*** by looking at the notes below.

Web services - Introduction

Ordinary websites, written in HTML, are very much geared towards the end user. The end user uses the HTML interface to perform particular tasks. However the end user is not the only possible consumer of information on the web. What about other applications? A ***web service*** is an application running on a web server which provides raw information to other applications. These other applications are clients to the web service because they make a connection to it and process the information sent back. Here are a few examples of web services and their clients:

- ***Flights:*** An airline provides flight times and a booking service available on its own website. However, other organisations might wish to make use of this. Third-party sites such as Expedia.com or travel agents may wish to provide this service themselves. In these cases, scripts on the Expedia.com web server, or a standalone booking application at a travel agent (which might be written in Java, for example) might connect to the airline's web service, and receive flight information back. The clients (Expedia and the travel agent application) could also contact the web service to perform bookings.

- ***Weather:*** a weather web service might make available current conditions in various cities around the world. Then, a website (e.g. a tourism website for London or Paris) could connect to the weather web service and display the current conditions in that city, retrieved from the web service but at the same time integrated seamlessly into the layout of its own site. Furthermore, a smartphone "app" could connect to the web service and show the weather for the user's current location within the app's interface.
- ***Mapping:*** electronic maps are typically stored in a database, which contains information about the roads and the latitude/longitude coordinates making up each point on the road (position on the earth, e.g. we are at approximately latitude 50.9 degrees North, longitude 1.4 degrees West, the Equator being latitude 0, the North Pole, 90 degrees North, and the Greenwich Meridian, longitude 0). A mapping web service could make available this mapping information, so that client websites could use the raw data sent back from the web service to draw a map in their own style and colour scheme. OpenStreetMap (http://www.openstreetmap.org) is an example of such a mapping web service.

How is information sent back from web services?

Web services obviously send back information to their clients, but what format is typically used? HTML is one possibility but is not considered a good idea because HTML contains not only data, but also page structure information (headings, paragraphs, tables etc). A client to the web service might wish to arrange the information in a different way. Also some web service clients are not websites at all, for example the Java application used by the travel agent in the above example, or the weather "app". We could still use HTML, however it would be tricky to write a ***parser*** - a program which interprets the data and extracts the relevant information from it - as HTML contains all manner of formatting and style information in addition to the data itself.

So what we want is a format which represents the data, and the data alone. There are a number of formats we can use including ***XML*** and ***JSON***.

JSON - JavaScript Object Notation is another format which uses JavaScript syntax to represent data.

With JSON we represent a single entity (such as a person, a song or a film) using curly brackets {} and we represent a collection of entities using square brackets []. (The basic idea is that data is transported across the web ***as a text representation of a JavaScript array or object***, but we will look at this in more detail later).

Here is an example of some data representing a series of students in JSON.

```
[
    {
      "name": "Tim Smith",
      "username": "ts282",
      "course": "Computer Studies"
    },

    {
      "name": "Jamie Bailey",
      "username": "jb139",
      "course": "Computer Studies"
    },

    {
      "name": "Deep Patel",
      "username": "dp061",
      "course": "Networks and Web Design"
    }
]
```

Note how each student is represented by curly brackets { and }, and the collection of students is represented by square brackets [and].

Generating JSON server-side

JSON can be generated server-side in PHP, from a set of database results. PHP (v5.2.0 onwards) provides a function, ***json_encode()***, to automatically generate JSON from PHP arrays or associative arrays (see last time for more on associative arrays). Here is an example to generate JSON from a PHP array:

```
$monthLengths = array ( 31, 28, 31, 30, 31, 30, 31, 31,
echo json_encode($monthLengths);
```

This will produce the JSON:

```
[ 31, 28, 31, 30, 31, 30, 31, 31, 30, 31, 30, 31 ]
```

Here is an example showing an associative array:

```
$obama = array ("name" => "Barack Obama",
                "age" => 56,
                "nationality" => "US",
            "job" => "US President 2008-16" );
```

```
echo json_encode($obama);
```

This would give the JSON:

```
{"name" : "Barack Obama", "age" : 56, "nationality" : "
```

and finally here is an example showing an array of associative arrays:

```
$politicians = array (

       array ("name" => "Barack Obama",
                 "age" => 56,
                 "nationality" => "US",
            "job" => "US President 2008-16" ),

       array ("name" => "Bernie Sanders",
                 "age" => 76,
                 "nationality" => "US",
            "job" => "US Presidential candidate 2016" )

                      );
echo json_encode($politicians);
```

This will produce this JSON:

```
[
    {"name" : "Barack Obama", "age" : 56, "nationality"
    {"name" : "Bernie Sanders", "age" : 76, "nationalit
     "job": "US Presidential candidate 2016" }
]
```

The rule is that PHP arrays will map to JSON/JavaScript arrays, and PHP associative arrays will map to JSON/JavaScript objects.

JSONView - Firefox JSON viewer plugin

The JSON output from server-side applications should be given a content type of ***application/json***; this will enable clients to recognise the content as JSON. However, by default, Firefox will not recognise JSON: therefore, you will not be able to easily test out a server-side script which generates JSON using the browser. To enable Firefox to recognise JSON you should install the

JSONView plugin, available here. This will format the JSON nicely in the browser as long as the content type has been set correctly, e.g. in PHP:

```
header("Content-type: application/json");
```

Exercise

Change your PHP script from last time to output JSON rather than HTML.

- First, remove all your code to output HTML from your PHP.
- Next, add the header() call:

  ```
  header("Content-type: application/json");
  ```

  to tell the browser that JSON is being returned.
- Finally, generate your JSON. You can do this very simply, you just have to remember that ***json_encode()*** can be used to encode arrays, associative arrays or arrays of arrays. Just ***one new line of code*** is needed!

To test, install the JSONView plugin, as described above from here (Firefox) , or here (Chrome)

Preparation for next week

Create a one-page "fan" website for your favourite artist. The site should have some CSS (enough to make it distinctive) and should include a short biography of the artist (get this from Wikipedia if necessary). **Next week, you will connect this site to your web service, retrieving all songs by that artist from the web service and displaying them on the site.**

You need to make sure you use an artist in the database. See here for a list of all the songs present.